TUGAS AKHIR – KI1502

PENGEMBANGAN MODEL OPTIMASI UNTUK PENJADWALAN DOKTER PADA KEADAAN DARURAT BERBASIS APLIKASI *MOBILE*

QONITA LUTHFIA SUTINO
NRP 5113100192

Dosen Pembimbing
Fajar Baskoro, S.Kom., MT.
Ahmad Saikhu, S.SI., MT.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI1502

PENGEMBANGAN MODEL OPTIMASI UNTUK PENJADWALAN DOKTER PADA KEADAAN DARURAT BERBASIS APLIKASI *MOBILE*

QONITA LUTHFIA SUTINO
NRP 5113100192

Dosen Pembimbing
Fajar Baskoro, S.Kom., MT.
Ahmad Saikhu, S.SI., MT.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESES – KI1502

DEVELOPMENT OF OPTIMIZATION MODEL FOR SCHEDULING DOCTOR TOWARD EMERGENCY CONDITION IN MOBILE-BASED APPLICATION

QONITA LUTHFIA SUTINO
NRP 5113100192

Supervisors
Fajar Baskoro, S.Kom., MT.
Ahmad Saikhu, S.SI., MT.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

PENGEMBANGAN MODEL OPTIMASI UNTUK PENJADWALAN DOKTER PADA KEADAAN DARURAT BERBASIS APLIKASI *MOBILE*

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Rekayasa Perangkat Lunak
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
QONITA LUTHFIA SUTINO
NRP: 5113 100 192

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Fajar Baskoro, S.Kom., MT. .....
(NIP 197404031999031002) (Pembimbing 1)

Ahmad Saikhu, S.SI., MT. .....
(NIP 197107182006041001) (Pembimbing 2)

SURABAYA
JANUARI, 2017

[Halaman ini sengaja dikosongkan]

PENGEMBANGAN MODEL OPTIMASI UNTUK PENJADWALAN DOKTER PADA KEADAAN DARURAT BERBASIS APLIKASI *MOBILE*

Nama Mahasiswa	**:**	**Qonita Luthfia Sutino**
NRP	**:**	**5113 100 192**
Jurusan	**:**	**Teknik Informatika FTIf-ITS**
Dosen Pembimbing 1	**:**	**Fajar Baskoro, S.Kom., MT.**
Dosen Pembimbing 2	**:**	**Ahmad Saikhu, S.SI., MT.**

Abstrak

Kejadian tak terduga, seperti bencana alam, seringkali menyebabkan kondisi darurat dan memakan korban. Korban bencana tentu perlu ditangani dengan cepat dan tepat oleh tenaga medis profesional, seperti dokter, sehingga nyawa korban dapat terselamatkan. Maka dari itu, dokter perlu ditugaskan sesuai dengan jadwal yang diatur menggunakan sistem yang praktis, seperti aplikasi mobile. Hal ini disebabkan tingginya mobilitas dalam menjalankan tugas medis pada wilayah bencana. Namun, kondisi darurat yang terjadi bersamaan dengan kondisi darurat di lokasi yang berbeda dan terbatasnya jumlah dokter, keahlian dokter, serta beragamnya keahlian yang diperlukan pada lokasi darurat merupakan kemungkinan yang dapat terjadi pada kondisi darurat. Untuk mengatasi masalah ini, aplikasi yang dibangun perlu menghasilkan solusi yang optimal.

Tujuan dibangunnya aplikasi mobile ini sebagai tugas akhir adalah untuk menerapkan model Graf Bipartite dengan Algoritma Ford-Fulkerson untuk mendapatkan dokter yang memenuhi kondisi darurat dan Integer Programming dengan metode penugasan optimal untuk Minimization Case. Selain itu, aplikasi ini juga ditujukan untuk diujicobakan pada studi kerawanan bencana, khususnya wilayah Jawa Timur.

Kata kunci: kondisi darurat, scheduling optimization model, ILP, mobile application

DEVELOPMENT OF OPTIMIZATION MODEL FOR SCHEDULING DOCTOR TOWARD EMERGENCY CONDITION IN MOBILE-BASED APPLICATION

Student Name	**:**	**Qonita Luthfia Sutino**
NRP	**:**	**5113 100 192**
Major	**:**	**Informatics Department FTIf – ITS**
Advisor I	**:**	**Fajar Baskoro, S.Kom., MT.**
Advisor II	**:**	**Ahmad Saikhu, S.SI., MT.**

Abstract

Unexpected phenomenon like disaster often causes emergency condition and involves victims. The victims have to be treated properly and immediately by professional medics so that their lives could be safe. To make it possible, there is a need to assign the medics or doctors, using a portable schedule system, like mobile application, considering the need of high mobility for performing medical duty. However, the problems are an emergency condition possibly occurs in a location coinciding with other emergency condition in different location and there are limited number of doctors, medical specializations, and some varieties of required qualifications in an emergency location. It means, the application is needed to result optimal solution to overcome the problems.

The purposes of developing this application in mobile platform as undergraduate theses are implementing Bipartite Graph model with Ford-Fulkerson Algorithm to get qualified doctors according the emergency condition and implementing Integer Programming with optimal assignment method for minimization case. Moreover, it is purposed to be tested, particularly, based on East Java regions disaster-prone study.

Keywords: kondisi darurat, scheduling optimization model, ILP, mobile application

KATA PENGANTAR

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT, yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"PENGEMBANGAN MODEL OPTIMASI UNTUK PENJADWALAN DOKTER PADA KEADAAN DARURAT BERBASIS APLIKASI *MOBILE*".**

Pengerjaan tugas akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan tugas akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS.

Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak, sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Ibunda dan Ayahanda, yang selalu mendoakan penulis dan mendukung setiap pilihan yang penulis ambil.
3. Bapak Fajar Baskoro, S.Kom., MT. selaku pembimbing I yang selalu memberikan motivasi dan membimbing penulis selama pengerjaan tugas akhir.
4. Bapak Ahmad Saikhu, S.SI., MT. selaku pembimbing II yang selama ini telah membantu dan membimbing penulis selama pengerjaan tugas akhir.
5. Bapak Dr.Eng Darlis Herumurti, S.Kom.,M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Dr. Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah banyak memberikan ilmu kepada penulis.

6. Mas Ramadhan R. P. yang sudah bersedia membagi ilmunya kepada penulis.
7. Teman-teman Badan Pengurus Harian Schematics HMTC ITS 2015 yang telah memberikan kesempatan penulis untuk mengembangkan *softskill* penulis.
8. Teman-teman Administrator Laboratorium RPL yang telah menjadi keluarga selama penulis menimba illmu di Teknik Informatika ITS.
9. Teman-teman Staf Departemen Pengembangan Profesi HMTC Berkarya 2014/2015 dan Departemen Pengembangan Profesi HMTC Optimasi 2015/2016.
10. Teman-teman angkatan 2013 yang telah membantu, berbagi ilmu, menjaga kebersamaan, dan memberi motivasi kepada penulis, serta adik-adik angkatan 2014 dan 2015 yang membuat penulis untuk selalu belajar.
11. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis menyadari bahwa tugas akhir ini masih memiliki banyak kekurangan, sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Januari 2017

Qonita Luthfia Sutino

DAFTAR ISI

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Bab ini menjelaskan garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan, batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Kejadian tidak terduga, seperti bencana alam, merupakan suatu kondisi darurat yang tidak jarang menimbulkan banyak korban. Untuk menanggulangi korban bencana alam tersebut, tentu diperlukan penanganan medis yang cepat dan tepat oleh tenaga profesional. Kondisi darurat seperti ini dapat terjadi kapanpun dan di manapun dalam suatu waktu yang bersamaan. Maka dari itu, dokter yang menangani kondisi darurat ini harus memiliki keahlian sesuai yang dibutuhkan dan juga harus memiliki mobilitas yang tinggi untuk terjun langsung ke lokasi darurat tersebut.

Namun, tidak jarang kondisi darurat yang terjadi pada suatu waktu di lokasi yang berbeda memerlukan keahlian dokter yang sama yang jumlahnya hanya cukup untuk menanggulangi kondisi darurat di salah satu lokasi saja. Dengan adanya keterbatasan jumlah dokter ini, diperlukan koordinasi penugasan dokter yang cepat dan tepat. Koordinasi penugasan dokter ini dapat berupa penjadwalan dokter yang dapat disusun oleh pihak tertentu, seperti manajemen rumah sakit, badan koordinasi penanggulangan bencana, atau Palang Merah Indonesia.

Penjadwalan dokter pada kondisi darurat ini memiliki beberapa variabel, yaitu kejadian darurat pada suatu lokasi, jumlah dokter dan keahlian masing-masing dokter, serta kebutuhan akan keahlian dokter tertentu di lokasi darurat tertentu. Dokter dan keahlian dipetakan ke dalam Graf *Bipartite*. Untuk menentukan dokter yang sesuai dengan kondisi yang dibutuhkan, digunakan Algoritma *Ford-Fulkerson*. Setelah pengelompokkan dokter dan kondisi sesuai, dilakukan penentuan jarak terpendek

dokter untuk menuju ke daerah dengan kondisi yang sudah dicocokkan sebelumnya. Hal ini dilakukan dengan menggunakan model *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case*. Hasil dari metode ini adalah jarak paling optimal yang dapat ditempuh dokter untuk ditugaskan ke daerah dengan kondisi yang telah disesuaikan sebelumnya.

Hasil yang diharapkan dari pengerjaan tugas akhir ini adalah berupa aplikasi perangkat lunak berbasis *mobile* yang mempermudah penugasan dokter ke daerah dengan kondisi yang dibutuhkan sesuai dengan klasifikasi keahlian dokter tersebut.

1.2. Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimana menerapkan model Graf *Bipartite* pada perancangan aplikasi *mobile* untuk menentukan dokter yang memenuhi kondisi?
2. Bagaimana menerapkan model *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case* pada perancangan aplikasi *mobile*, sehingga dapat menghasilkan penugasan dokter paling optimal yang memenuhi kondisi daerah tersebut?

1.3. Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Sistem perangkat lunak dibangun dengan menggunakan bahasa Java dan aplikasi Android Studio.
2. Inputan perangkat lunak meliputi daftar dokter dan daftar daerah yang dipilih dan sesuai dengan data pada *database*.
3. Hasil yang dikeluarkan adalah daftar dokter yang memenuhi kondisi daerah yang dituju dengan jarak paling optimal, total jarak tempuh dokter, dan daerah yang dituju.

4. Satu lokasi bencana tertentu hanya boleh ditangani oleh dokter dengan kualifikasi yang sesuai dengan keahlian yang dibutuhkan pada lokasi bencana.
5. Uji coba aplikasi ini menggunakan data *dummy* (bukan data sebenarnya) pada *database*.

1.4. Tujuan

Tugas akhir ini mempunyai beberapa tujuan, yaitu sebagai berikut:

1. Membangun sistem penjadwalan/penugasan dokter pada keadaan darurat (bencana) dengan metode Graf *Bipartite*, Algoritma *Ford-Fulkerson*, dan *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case* yang dapat diakses melalui perangkat *mobile*.
2. Melakukan uji coba aplikasi perangkat lunak ini pada studi kerawanan bencana, khususnya di wilayah Jawa Timur.

1.5. Manfaat

Tujuan dari pembuatan tugas akhir ini adalah:

1. Menghasilkan keputusan yang tepat dalam penentuan penugasan dokter dalam mewujudkan layanan panggilan dokter pada keadaan gawat darurat yang didasarkan keahlian dokter yang dibutuhkan secara *real time*. Dengan begitu, situasi gawat darurat di suatu daerah dapat ditanggulangi secepat mungkin.

1.6. Metodologi Pembuatan Tugas Akhir

Tahapan-tahapan yang dilakukan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Penyusunan proposal tugas akhir.

   Proposal tugas akhir ini terdiri dari deskripsi pendahuluan yang menjabarkan latar belakang dan rumusan masalah yang mendasari dibangunnya aplikasi ini, batasan masalah dalam pembangunan aplikasi ini, serta tujuan dan manfaat yang diharapkan dapat dicapai dengan dibangunnya aplikasi ini.

Selain itu, pada proposal tugas akhir ini juga terdapat tinjauan pustaka yang menjelaskan teori-teori yang menjadi dasar pembuatan tugas akhir ini, yaitu Keadaan Darurat (Bencana), Optimasi Permasalahan Penugasan Dokter Menggunakan Representasi Graf *Bipartite* Berbobot, Penggunaan Algoritma *Ford-Fulkerson* dalam Pengelompokkan Dokter Berdasarkan Keahlian Dokter, dan Pemodelan Penugasan Dokter secara Optimal Menggunakan *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case*.

2. Studi literatur

   Studi literatur yang dilakukan pada perancangan aplikasi *mobile* sebagai pengerjaan tugas akhir ini adalah mengenai implementasi himpunan dokter dan himpunan keahlian dokter ke dalam Graf *Bipartite* dan Algoritma *Ford-Fulkerson* untuk mendapatkan penugasan dokter yang sesuai dengan kondisi yang dibutuhkan. Selain itu, juga dilakukan studi literatur mengenai implementasi model *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case* untuk memperoleh jarak penugasan dokter ke daerah bencana yang paling optimal. Sehingga, studi literatur ini dapat diterapkan pada perancangan aplikasi *mobile* berdasarkan hasil penugasan dokter yang optimal yang memenuhi kondisi darurat daerah tertentu.

3. Analisis dan desain perangkat lunak

   Tahap ini meliputi perumusan kebutuhan fungsional, kebutuhan non-fungsional, kasus penggunaan, diagram aktivitas, diagram kelas, diagram sekuens, rancangan antarmuka pengguna untuk akun admin dan dokter, serta pembuatan rancangan basis data.

4. Implementasi perangkat lunak

Aplikasi ini diimplementasikan dengan menggunakan kakas bantu :

1. Sistem operasi Android dengan spesifikasi minimal Android 4.0 (*Ice Cream Sandwich*).
2. Bahasa pemrograman Java.
3. IDE Android Studio.
4. Database MySQL.
5. Kerangka kerja *web service* CodeIgniter.
6. *Hosting web* yang menjadi acuan dalam perancangan aplikasi ini dari dewaweb.com
7. Postman, kakas bantu untuk menguji integrasi *web service* dengan sistem yang sudah dibangun.
8. Sublime Text sebagai *text editor* dalam pengerjaan *web service.*

5. Pengujian dan evaluasi

   Pengujian dan evaluasi aplikasi perangkat lunak hasil dari tugas akhir ini diujicobakan pada studi kerawanan bencana, khususnya di wilayah Jawa Timur, dengan menggunakan *database* kerawanan bencana dan tenaga medis yang ada.

6. Penyusunan buku tugas akhir

   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

   1. Pendahuluan
      a. Latar Belakang
      b. Rumusan Masalah
      c. Batasan Masalah
      d. Tujuan
      e. Manfaat
      f. Metodologi Pembuatan Tugas Akhir
      g. Sistematika Penulisan Laporan Tugas Akhir
   2. Tinjauan Pustaka

3. Analisis dan Perancangan Sistem
4. Pengujian dan Evaluasi
5. Kesimpulan dan Saran
6. Daftar Pustaka

1.7. Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, dan manfaat dari pembuatan tugas akhir. Selain itu metodologi yang digunakan dan sistematika penulisan laporan akhir juga merupakan bagian dari bab ini.

Bab II Tinjauan Pustaka

Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini berisi tentang analisis permasalahan, deskripsi umum sistem, spesifikasi kebutuhan perangkat lunak, lingkungan perancangan, perancangan arsitektur sitem, diagram kelas, dan struktur data.

Bab IV Implementasi

Bab ini membahas implementasi dari desain yang telah dibuat pada bab sebelumnya. Penjelasan berupa kode sumber yang digunakan untuk proses implementasi.

Bab V Pengujian dan Evaluasi

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

Bab VI Kesimpulan dan Saran

Bab ini menjelaskan kemampuan perangkat lunak dengan melakukan pengujian kebenaran dan pengujian kinerja dari sistem yang telah dibuat.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan teori-teori yang berkaitan dengan pembangunan aplikasi *mobile* penugasan dokter yang diajukan untuk tugas akhir ini. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap perangkat lunak yang dibuat dan berguna sebagai penunjang dalam pengembangan perangkat lunak.

2.1 Android

Android adalah sistem operasi perangkat bergerak yang dikembangkan Google dan berbasis Linux yang dirancang dengan layar sentuh. Pertama kali, sistem operasi Android dikembangkan oleh Android, Inc hingga pada bulan Juli 2005 bergabung dengan Google. Android resmi dirilis pada tanggal 5 November 2007 dan setelah itu Android terus mengalami perkembangan hingga saat ini [1]. Dibandingkan dengan sistem operasi *smartphone* lainnya, Android sudah digunakan di Indonesia dengan pengguna sebanyak 93,6 persen, bahkan di tingkat asia tenggara, pengguna Android di Indonesia merupakan pengguna Android paling banyak, yaitu sebesar 32,7 persen [2]. Hal ini merupakan salah satu hal yang menjadi pertimbangan penulis dalam membangun aplikasi *mobile* ini dengan sistem operasi Android.

2.2 REST Web Service

REST [3] yang merupakan singkatan dari *Representational State Transfer* adalah standar dalam arsitektur web yang menggunakan Protocol HTTP untuk pertukaran data. Pertama-tama, REST *server* menyediakan jalur untuk akses *resource* atau data, sedangkan REST *client* melakukan akses *resource* dan kemudian menampilkan atau menggunakannya. *Resource* yang dihasilkan sebenarnya berupa teks, namun formatnya bisa bermacam-macam tergantung keinginan *developer*, umumnya adalah JSON dan XML.

Dalam mengakses sebuah *resource*, REST juga menggunakan konsep URI (Uniform Resource Identifiers) dengan *method* standar yang digunakan adalah GET. Berikut ini *method-method* yang mendukung REST:

- GET, cocok untuk *resource* yang hanya perlu dibaca saja (*read only*)
- PUT, cocok digunakan untuk membuat *resource* baru.
- DELETE, cocok digunakan untuk menghapus suatu *resource*.
- POST, cocok digunakan untuk update suatu *resource*.

Cara kerja REST adalah sebuah *client* mengirimkan sebuah data atau *request* melalui HTTP Request dan kemudian *server* merespon melalui HTTP Response.
Komponen dari HTTP Request adalah:

- Verb, HTTP *method* yang digunakan misalnya GET, POST, DELETE, PUT dll.
- URI, Uniform Resource Identifier (URI) untuk mengidentifikasikan lokasi *resource* pada *server*.
- HTTP Version, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
- Request Header, berisi metadata untuk HTTP Request. Contohnya adalah *type client/browser*, format yang didukung oleh *client*, format dari *body* pesan, pengaturan *cache*, dll.
- Request Body, konten dari data.

Komponen dari HTTP Response adalah:

- Status/Response Code, mengindikasikan status *server* terhadap *resource* yang diminta. Misal: 404, artinya *resource* tidak ditemukan dan 200 response OK.
- HTTP Version, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.

- Response Header, berisi metadata untuk HTTP Response. Contoh, type server, panjang content, tipe content, waktu response, dll.
- Response Body, konten dari data yang diberikan.

2.3 MySQL

MySQL [4] adalah sebuah perangkat lunak sistem manajemen basis data SQL di bawah lisensi GPL (General Public License). MySQL mendukung operasi basis data transaksional dan non-transaksional. Berikut ini adalah beberapa keistimewaan pada MySQL:

- Portabilitas, yaitu MySQL mampu berjalan stabil pada berbagai sistem operasi.
- Aplikasi *open source*, sehingga dapat digunakan secara gratis di bawah lisensi GPL.
- *Performance tuning*, yaitu menangani query sederhana dengan cepat.
- Skalabilitas dan Pembatasan, yaitu MySQL dapat menangani basis data dalam skala besar dengan *record* lebih dari 50 juta dan 60 ribu tabel, serta 5 miliar baris.
- Struktur tabel MySQL lebih fleksibel dalam menangani *ALTER TABLE* dibandingkan dengan basis data lainnya.

Selain itu, MySQL memiliki kelebihan, yaitu dapat diintegrasikan dengan beberapa bahasa pemrograman .Net, Java, Python, dan Perl.

2.4 Volley

Volley adalah *library* HTTP yang mempermudah dan mempercepat *networking* pada aplikasi Android [5]. Volley yang diterapkan pada aplikasi ini terdiri dari:

- Mengirim *request* yang merupakan penggunaan Volley dengan membuat RequestQueue dan menyampaikannya objek Request [6].

- Membuat Request standar dengan menggunakan StringRequest yang menentukan URL dan menerima *string*, serta JsonRequest (JsonObjectRequest dan JsonArrayRequest) yang menentukan URL dan memperoleh objek JSON atau *array* sebagai respon [7].

2.5 Keadaan Darurat (Bencana)

Keadaan darurat dapat terjadi dari adanya bencana alam yang tak terelakkan. Bencana alam dapat terjadi kapanpun dan di manapun dalam waktu yang bersamaan. Kerusakan infrastruktur dan korban yang berjatuhan dari adanya bencana alam adalah keadaan darurat yang harus ditangani secepat dan setepat mungkin demi keselamatan korban bencana.

Mitigasi bencana adalah serangkaian upaya untuk mengurangi risiko bencana, baik melalui pembangunan fisik maupun penyadaran dan peningkatan kemampuan menghadapi ancaman bencana [8]. Hal ini dapat dilakukan dengan mendeteksi keadaan bumi untuk mengetahui bencana apa saja yang rawan pada suatu daerah di wilayah Indonesia, khususnya daerah Jawa Timur. Penerapan *early warning system* ini diharapkan dapat meningkatkan sikap waspada, siaga, hingga meninggalkan lokasi (evakuasi) bila keadaan sudah membahayakan pada masyarakat.

Tujuan dari adanya mitigasi bencana adalah mengurangi kerugian-kerugian, terutama kerugian seperti risiko kematian dan cedera penduduk pada saat terjadinya bahaya pada masa mendatang [9]. Tujuan sekunder mitigasi bencana mencakup pengurangan kerusakan dan kerugian-kerugian ekonomi yang ditimbulkan terhadap infrastruktur sektor publik yang dapat mempengaruh masyarakat luas. Maka, mitigasi bencana tidak hanya membantu masyarakat, tetapi juga pembangunan dari bencana yang tidak dapat diduga.

2.6 Optimasi Permasalahan Penugasan Dokter Menggunakan Representasi Graf Bipartite Berbobot

Graf *Bipartite* adalah graf yang simpulnya dapat dipisah menjadi dua himpunan [10]. Dalam menentukan penugasan dokter ini, diperlukan dua informasi penting, yaitu informasi dokter dengan keahlian dokter dan informasi kondisi darurat dengan keahlian medis yang diperlukan. Sehingga, masing-masing informasi direpresentasikan ke dalam Graf *Bipartite*.

2.7 Penggunaan Algoritma Ford-Fulkerson dalam Pengelompokkan Dokter Berdasarkan Keahlian Dokter

Algoritma *Ford-Fulkerson* [11] digunakan untuk memecahkan permasalahan *network flow* dalam pemasangan maksimal dalam Graf *Bipartite* yang merupakan representasi dari kondisi dan dokter. *Network flow* tersebut terdiri dari *source* s yang berhubungan dengan node kondisi dan *sink* t yang berhubungan dengan node dokter, serta *edge*. *Edge* yang menghubungkan antara *source* dengan kondisi diberi bobot sebesar banyaknya dokter, sedangkan *edge* antara node dokter dengan *sink* diberi bobot sebesar jumlah kondisi. Sehingga, dokter dikelompokkan sesuai dengan keahlian yang memenuhi kondisi. Secara umum, langkah-langkah pada algoritma *Ford-Fulkerson* adalah sebagai berikut, di mana graf $G(V, E)$ yang setiap *edge* nya dimulai dari u ke v dengan kapasitas $c(u, v)$ dan *flow* $f(u, v) = 0$:

1. Inisialisasi nilai f menjadi 0, $f(u, v) \leftarrow 0$ untuk setiap *edge*(u,v)
2. Melakukan perulangan proses di atas selama terdapat jalur p dari s menuju t (*augmenting path* p) di mana $c_f(u, v) > 0$ untuk semua (u, v) adalah himpunan p. *Augmenting path* adalah jalur dengan kapasitas yang sesuai [].
   a. $c_f(p) = \min\{c_f(u, v) | (u, v) E\ p\}$
   b. Untuk setiap *edge* (u, v) E p
      i. $f(u, v) \leftarrow f(u, v) + c_f(p)$ (mengirimkan aliran melalui jalur)

ii. $f(u,v) \leftarrow f(u,v) - c_f(p)$(aliran mungkin akan kembali)

2.8 Model *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case*

Setelah diperoleh daftar dokter yang memenuhi kondisi yang dibutuhkan, dilakukanlah penugasan dokter dengan model *Integer Programming* dengan masukan data berupa jarak dokter ke daerah yang memiliki kondisi darurat. Batasan pada pemodelan ini adalah tiap dokter hanya ditugaskan pada tepat satu kondisi darurat, begitu juga sebaliknya. Total jarak paling optimal yang dipilih adalah total jarak terkecil, sehingga penulis menggunakan Metode Penugasan Optimal untuk *Minimization Case* [12]. Berikut ini adalah penjelasan langkah-langkah *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case*:

1. Menyusun data dalam matriks penugasan dengan baris sebagai kondisi yang diasumsikan sebagai daerah dan kolom sebagai dokter. Misalkan, tabel jarak antara daerah dan dokter yang sudah memenuhi kondisi sudah diperoleh seperti pada Tabel 2.1. Dengan begitu, matriks penugasan dapat digambarkan seperti Tabel 2.2.

Tabel 2.1 Data Jarak setelah Implementasi Algoritma Ford-Fulkerson

Dokter	**Daerah**		
	K1	K2	K3
D1	999	999	999
D2	999	**40**	**6**
D3	999	**2**	**14**
D4	999	999	999
D5	**3**	999	999
D6	999	999	999

Tabel 2.2 Matriks Penugasan

	D1	D2	D3	D4	D5	D6
K1	999	999	999	999	**3**	999
K2	999	**40**	**2**	999	999	999
K3	999	**6**	**14**	999	999	999

2. Membuat dua kolom dengan kolom 1 merepresentasikan daerah dan kolom 2 merepresentasikan dokter. Pada kolom satu berisi nama daerah bencana sebanyak daerah bencana yang diperoleh. Kemudian, mencari jarak minimum pada setiap baris matriks dan menuliskan indeks dokter dengan jarak minimum tersebut di bawah kolom 2 seperti pada Tabel 2.3.

Tabel 2.3 Kolom Daerah dan Dokter

Daerah	Dokter
K1	D5
K2	D3
K3	D2

3. Untuk setiap daerah, jika terdapat dokter yang unik (indeks dokter yang terdapat pada kolom 2 hanya satu), dokter unik tersebut ditugaskan pada daerah ini. Dengan begitu, solusi optimal untuk daerah ini sudah diperoleh. Seperti pada Tabel 2.3, setiap daerah K1, K2, dan K3 memiliki dokter yang berbeda, sehingga D5 pada K1, D3 pada K2, dan D2 pada K3 merupakan dokter yang unik.
4. Menetapkan dokter yang unik ke daerah yang sudah sesuai tersebut. Kemudian, menghapus baris daerah yang sudah ditugaskan tersebut pada kedua kolom (kolom daerah dan dokter) dan matriks. Jika belum semua daerah memperoleh solusi, dilanjutkan ke langkah ke-5.
5. Cari baris pada kolom 1 (daerah) yang memiliki indeks dokter yang sama pada kolom 2 nya. Kemudian, bandingkan selisih jarak minimum tersebut dengan jarak minimum lain pada daerah tersebut. Selisih jarak

minimum yang paling besar adalah jarak minimum yang diambil pada daerah tersebut. Dengan begitu, dokter tersebut ditugaskan pada daerah yang memiliki selisih minimum jarak paling besar. Lalu, baris daerah pada kedua kolom dan matriks dihapus. Misal, suatu matriks dan kedua kolom memiliki data seperti pada Tabel 2.4 dan Tabel 2.5.

Tabel 2.4 Matriks Penugasan ketika Indeks Dokter Sama

	D1	D2	D3	D4	D5	D6
K1	999	999	**3**	999	999	999
K2	999	999	**2**	999	999	999

Tabel 2.5 Kolom Daerah dan Dokter ketika Indeks Dokter Sama

Daerah	Dokter
K1	D3
K2	D3

Selisih jarak seperti pada Tabel 2.6 menghasilkan D3 ditugaskan ke K2, karena selisih jarak minimum K2 lebih besar dari jarak minimum K1.

Tabel 2.6 Selisih Jarak Minimum

Daerah	Selisih Jarak
K1	999 – 3 = 996
K2	999 – 2 = 997

6. Mengulang langkah ke-4 sampai ke-5 hingga semua daerah sudah memperoleh penugasan dokter.
7. Menghitung total jarak yang diperoleh setelah solusi optimal keseluruhan didapatkan.

2.9 Model *Integer Programming* dengan Metode Penugasan Optimal untuk *Unbalanced Assignment Problem*

Dalam penanganan medis, tidak dapat dipungkiri akan adanya kemungkinan dokter yang tersedia untuk melakukan penugasan tidak sebanyak kondisi darurat yang perlu ditangani. Untuk mengatasi kemungkinan ini, digunakan Metode Penugasan Optimal untuk *Unbalanced Assignment Problem* [12], di mana permintaan untuk penanganan kondisi darurat lebih banyak daripada dokter yang tersedia. *Integer Programming* dengan Metode Penugasan Optimal untuk *Unbalanced Assignment Problem* ini terdiri dari beberapa langkah berikut:

1. Menyusun data dalam matriks penugasan dengan baris sebagai kondisi yang diasumsikan sebagai daerah dan kolom sebagai dokter dengan penambahan kolom *dummy* dokter yang berisi data jarak sama dengan 0 agar matriks seimbang. Misalkan, tabel jarak antara daerah dan dokter yang sudah memenuhi kondisi sudah diperoleh seperti pada Tabel 2.7. Dengan begitu, matriks penugasan dapat digambarkan seperti pada Tabel 2.8.

Tabel 2.7 Data Jarak setelah Implementasi Algoritma Ford-Fulkerson

Dokter	**Daerah**				
	K1	K2	K3	K4	K5
D1	**40**	999	**6**	**10**	999
D2	999	999	**6**	**4**	**15**
D3	999	**2**	**14**	999	999
D4	**3**	999	999	999	**20**

Tabel 2.8 Matriks Penugasan

	D1	D2	D3	D4	D5
K1	**40**	999	999	**3**	0

K2	999	999	**2**	999	0
K3	**6**	**6**	**14**	999	0
K4	**10**	**4**	999	999	0
K5	999	**15**	999	**20**	0

2. Membuat dua kolom dengan kolom 1 merepresentasikan daerah dan kolom 2 merepresentasikan dokter. Pada kolom satu berisi nama daerah bencana sebanyak daerah bencana yang diperoleh. Kemudian, mencari jarak minimum dan tidak sama dengan 0 pada setiap baris matriks dan menuliskan indeks dokter dengan jarak minimum tersebut di bawah kolom 2 seperti pada Tabel 2.9.

Tabel 2.9 Kolom Daerah dan Dokter

Daerah	Dokter
K1	D4
K2	D3
K3	D1, D2
K4	D2
K5	D2

3. Untuk setiap daerah, jika terdapat dokter yang unik (indeks dokter yang terdapat pada kolom 2 hanya satu), dokter unik tersebut ditugaskan pada daerah ini. Dengan begitu, solusi optimal untuk daerah ini sudah diperoleh. Berdasarkan pada Tabel 2.9, setiap daerah K1 dan K2 memiliki dokter yang berbeda, sehingga D4 pada K1 dan D3 pada K2 merupakan dokter yang unik.
4. Menetapkan dokter yang unik ke daerah yang sudah sesuai tersebut. Kemudian, menghapus baris daerah yang sudah ditugaskan tersebut pada kedua kolom (kolom daerah dan dokter) dan matriks. Jika belum semua daerah memperoleh solusi, dilanjutkan ke langkah ke-5.

5. Cari baris pada kolom 1 (daerah) yang memiliki indeks dokter yang sama pada kolom 2 nya. Kemudian, bandingkan selisih jarak minimum tersebut dengan jarak minimum lain pada daerah tersebut. Selisih jarak minimum yang paling besar adalah jarak minimum yang diambil pada daerah tersebut. Dengan begitu, dokter tersebut ditugaskan pada daerah yang memiliki selisih minimum jarak paling besar. Lalu, baris daerah pada kedua kolom dan matriks dihapus. Seperti pada Tabel 2.10 dan Tabel 2.11, daerah K3, K4, dan K5 memiliki indeks dokter yang sama, yaitu D2.

Tabel 2.10 Matriks Penugasan ketika Indeks Dokter Sama

	D1	D2	D5
K3	**6**	**6**	0
K4	999	**4**	0
K5	999	**15**	0

Tabel 2.11 Kolom Daerah dan Dokter ketika Indeks Dokter Sama

Daerah	**Dokter**
K3	D1, **D2**
K4	**D2**
K5	**D2**

Selisih jarak seperti pada Tabel 2.12 menghasilkan D2 ditugaskan ke K4, karena selisih jarak minimum K4 lebih besar dari jarak minimum K3 dan K5.

Tabel 2.12 Selisih Jarak Minimum

Daerah	**Selisih Jarak**
K3	999 – 6 = 993
K4	999 – 4 = 995
K5	999 – 15 = 984

6. Mengulang langkah ke-4 sampai ke-5 hingga solusi optimal penugasan dokter diperoleh. Jika tersisa satu kolom dokter (selain kolom dokter *dummy*) pada matriks, maka dokter ditugaskan ke daerah dengan jarak terpendek (bukan jarak pada kolom dokter *dummy*). Sedangkan daerah sisanya memperoleh solusi penanganan oleh dokter *dummy*. Berdasarkan Tabel 2.13 dan Tabel 2.14, setelah D2 ditugaskan ke K4, tersisa K3 dan K5 pada matriks penugasan serta pada kolom daerah dan dokter.

Tabel 2.13 Matriks Penugasan ketika Indeks Dokter Sama

	D1	D5
K3	**6**	0
K5	999	0

Tabel 2.14 Kolom Daerah dan Dokter ketika Indeks Dokter Sama

Daerah	**Dokter**
K3	D1
K5	D1

Karena pada tahap ini hanya tersisa satu kolom dokter (tidak termasuk kolom dokter *dummy*), K3 ditangani oleh D1. Hal ini disebabkan jarak D1 ke K3 lebih kecil daripada D1 ke K5. Dengan begitu, K5 ditangani oleh D5.

7. Menghitung total jarak yang diperoleh setelah solusi optimal didapatkan. Pada
8. Tabel 2.15, diperoleh solusi penugasan di mana K5 tidak memperoleh penanganan medis, karena dokter yang diperoleh adalah D5 yang merupakan kolom dokter *dummy* pada matriks penugasan.

Tabel 2.15 Total Jarak Penugasan

Daerah	Dokter	Jarak
K1	D4	3
K2	D3	2
K3	D1	6
K4	D2	4
K5	D5	0
Total		**15**

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas analisis kebutuhan dan rancangan yang akan digunakan untuk membangun perangkat lunak yang diajukan sebagai tugas akhir.

3.1 Analisis

Tahap analisis dibagi menjadi beberapa bagian, antara lain analisis permasalahan, deskripsi umum sistem, dan spesifikasi kebutuhan perangkat lunak.

3.1.1 Analisis Permasalahan

Bencana alam seringkali terjadi tanpa diduga sebelumnya dan dapat terjadi pada lokasi tertentu. Bahkan, tidak menutup kemungkinan, bencana alam dapat terjadi pada beberapa daerah secara bersamaan. Dengan begitu, bencana alam adalah kondisi darurat yang harus ditangani dengan cepat dan tepat untuk meminimalisasi jumlah korban yang berjatuhan. Penanganan medis pada kondisi darurat ini memerlukan keahlian dokter tertentu yang harus disesuaikan dengan keahlian yang diperlukan pada kondisi darurat tersebut. Hal ini dilakukan agar penanganan medis dapat dilakukan dengan efektif dan efisien. Untuk mewujudkannya, diperlukan penjadwalan dokter.

Dalam melakukan penjadwalan dokter, diperlukan beberapa variabel, yaitu kejadian darurat pada suatu lokasi, jumlah dokter dan keahlian masing-masing dokter, serta kebutuhan akan keahlian dokter tertentu di lokasi darurat tertentu. Namun, kemungkinan adanya hal seperti kondisi darurat dapat terjadi bersamaan dengan kondisi darurat di lokasi yang berbeda dan terbatasnya jumlah dokter, keahlian dokter, serta beragamnya keahlian yang diperlukan pada lokasi darurat dapat menghambat penanganan bencana secara tepat. Selain itu, tingginya mobilitas yang diperlukan dalam melakukan penanganan bencana menyebabkan penjadwalan dokter perlu dilakukan secara cepat.

Oleh karena itu, dalam melakukan penjadwalan dokter ini diperlukan aplikasi yang mudah diakses dan memberikan hasil keputusan yang optimal dalam menugaskan dokter yang sesuai ke daerah bencana yang berada pada kondisi darurat. Aplikasi tersebut berupa aplikasi *mobile* yang dapat diakses di mana saja dan kapan saja. Selain itu, keputusan yang dihasilkan berupa total jarak dokter yang optimal untuk menangani kondisi darurat ke daerah bencana yang ada. Dengan begitu, dokter dengan jarak perjalanan ke daerah bencana yang lebih kecil dapat menangani kondisi darurat di daerah tersebut lebih cepat daripada dokter dengan jarak perjalanan ke daerah bencana yang lebih besar.

Untuk menentukan keahlian dokter dan keahlian yang diperlukan pada kondisi darurat yang cocok, dokter dengan keahliannya dan kondisi darurat dengan keahlian yang diperlukan dimodelkan dengan Graf *Bipartite* dan menggunakan Algoritma *Ford-Fulkerson*. Kemudian, jarak terpendek ditentukan yang paling optimal menggunakan model *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case*. Dengan begitu, hasil yang diperoleh adalah berupa jarak yang paling optimal yang dapat ditempuh oleh tiap dokter yang memiliki keahlian yang cocok dengan keahlian yang ada pada tiap kondisi darurat.

Dalam peggunaan aplikasi ini, pengguna berhakakses admin terlebih dahulu menjadwalkan penugasan. Untuk mengetahui konfirmasi penugasan yang sudah dijadwalkan, admin perlu memantau aplikasi ini agar admin dapat mengecek dokter mana saja yang sudah memberikan konfirmasi penugasan, baik berupa persetujuan penugasan, penolakan penugasan, maupun konfirmasi penugasan selesai. Sedangkan pada sisi pengguna berhakakses dokter, aplikasi mobile ini perlu dipantau oleh dokter agar dokter dapat mengetahui penugasan apa saja yang dijadwalkan padanya. Setelah dokter memperoleh jadwal penugasan, dokter dapat memberikan konfirmasi menyetujui atau menolak penugasan tersebut. Jika penugasan sudah disetujui, dokter dapat memberikan konfirmasi penugasan selesai.

3.1.2 Modifikasi Algoritma Ford-Fulkerson

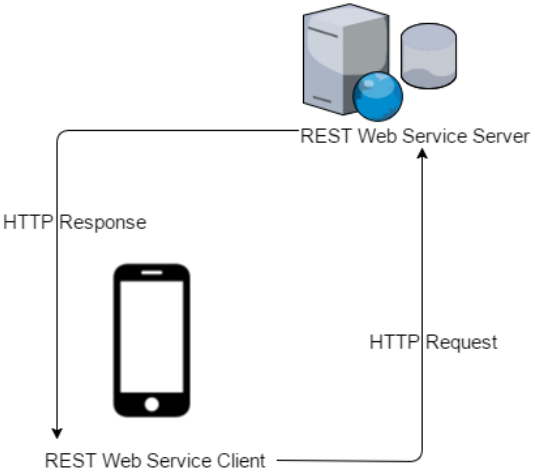
Algoritma *Ford-Fulkerson* adalah algoritma yang dapat digunakan untuk pemasangan maksimal pada Graf *Bipartite*. Pada sistem ini, algoritma *Ford-Fulkerson* mendukung pemasangan maksimal antara himpunan dokter yang ada dengan himpunan kondisi daerah bencana. Dengan begitu, penanganan medis daerah bencana dapat ditangani oleh dokter yang memiliki keahlian yang sesuai dengan keahlian yang diperlukan pada daerah bencana. Namun, untuk memperoleh hasil yang lebih optimal dengan mempertimbangkan jarak dokter ke daerah bencana, algoritma *Ford-Fulkerson* perlu untuk dimodifikasi. Jika menggunakan algoritma *Ford-Fulkerson* tanpa modifikasi, hasil yang diperoleh adalah pemasangan maksimal dokter yang memenuhi kondisi bencana tanpa memperhatikan jarak yang harus ditempuh dokter ke daerah bencana.

Modifikasi ini dilakukan dengan mengecek apakah keahlian yang diperlukan dalam penanganan medis suatu daerah bencana dimiliki oleh dokter yang ada. Hasil yang diperoleh dari modifikasi algoritma *Ford-Fulkerson* ini adalah data jarak dokter ke daerah bencana yang sudah diseleksi di mana keahlian dokter memenuhi keahlian yang dibutuhkan daerah bencana. Modifikasi pada algoritma *Ford-Fulkerson* untuk diterapkan pada aplikasi ini dijelaskan melalui *pseudocode* berikut ini:

```
// total_dokter adalah jumlah dokter
//daerah_bencana adalah array data daerah yang
terdiri dari jarak, rawan bencana, dan keahlian yang
dibutuhkan
for i ← 0 to total_dokter
  for i ← 0 to daerah_bencana.length
    //daftar_keahlian adalah array keahlian yang
dimiliki dokter
    //b_keahlian adalah string keahlian yang
dibutuhkan
```

```
        max ← 9999
        count ← 0
        for dk ← 0 to daftar_keahlian.length
          if b_keahlian contains in string of
    daftar_keahlian[dk]
            count++
        if count ← butuh_keahlian.length
          jarak_selected[i][j] ← jarak
        else
          jarak_selected[i][j] ← max
```

3.1.3 Deskripsi Umum Sistem



Gambar 3.1 Arsitektur Sistem

Arsitektur sistem aplikasi *mobile* ini digambarkan seperti pada Gambar 3.1. Pengguna sistem ini akan mengirim HTTP Request berupa Verb (GET/POST) + URL untuk mengakses data secara *real time*. Implementasi Graf *Bipartite*, Algoritma *Ford-Fulkerson*, dan *Integer Programming* dengan Metode Penugasan

Optimal untuk *Minimization Case* dijalankan pada *client* setelah memperoleh data dari *server*.

Dalam penggunaan aplikasi ini, sistem dibagi menjadi dua hak akses, yaitu admin dan dokter. Admin memiliki kewenangan untuk mengelola data pada sistem berupa melihat, menambah, mengedit, dan menghapus data yang ada pada sistem, serta mengatur penugasan dokter. Saat melakukan penugasan dokter ini, admin memilih dokter mana saja yang perlu ditugaskan dan daerah mana saja yang sedang mengalami kondisi darurat. Setelah itu, dokter akan memperoleh jadwal yang sudah diperoleh dari admin. Dokter dapat melakukan konfirmasi terlebih dahulu pada sistem untuk melakukan penugasan tersebut. Jika penugasan disetujui pada aplikasi dan selesai dilaksanakan, dokter dapat memberikan konfirmasi selesai melakukan penanganan medis dengan menekan tombol "selesai" pada aplikasi. Dengan begitu, penugasan dokter selesai.

3.1.4 Spesifikasi Kebutuhan Perangkat Lunak

Sesuai dengan cakupan perangkat lunak yang telah dijelaskan pada bagian deskripsi umum sistem, dibutuhkan adanya spesifikasi perangkat lunak agar dapat memberikan solusi dari permasalahan yang diberikan dan dapat bekerja dengan baik dalam mengakomodasi kebutuhan. Diharapkan dengan adanya spesifikasi ini dapat menyesuaikan kebutuhan-kebutuhan pengguna. Spesifikasi kebutuhan perangkat lunak adalah penjelasan mengenai kebutuhan sistem yang diinginkan pelanggan atau klien dalam bentuk tulisan. Spesifikasi kebutuhan perangkat lunak pada tugas akhir ini terdiri dari kebutuhan fungsional, kebutuhan non-fungsional, aktor, dan kasus penggunaan.

3.1.3.1. Kebutuhan Fungsional

Tabel 3.1 Kebutuhan Fungsional

No	Kebutuhan Fungsional	Deskripsi
1	Menangani kelola data	Menampilkan data daerah,

	daerah	melakukan proses penambahan, pengeditan, serta penghapusan data daerah
2	Menangani kelola data bencana	Menampilkan data bencana, melakukan proses penambahan, pengeditan, serta penghapusan data bencana
3	Menangani kelola data keahlian	Menampilkan data keahlian, melakukan proses penambahan, pengeditan, serta penghapusan data keahlian
4	Menangani kelola data rumah sakit	Menampilkan data rumah sakit, melakukan proses penambahan, pengeditan, serta penghapusan data rumah sakit
5	Menangani kelola data dokter	Menampilkan data dokter, melakukan proses penambahan, pengeditan, serta penghapusan data dokter
6	Menangani kelola data jarak penugasan	Menampilkan data daerah, melakukan proses penambahan dan pengeditan data jarak penugasan
7	Menangani penugasan dokter tertentu ke daerah bencana yang berada pada kondisi darurat	Menampilkan hasil penugasan dokter ke daerah bencana yang optimal yang ditunjukkan dengan jarak dokter ke daerah bencana yang optimal
8	Menangani penugasan ulang dokter	Melakukan proses untuk memperoleh dokter pengganti pada suatu proses penugasan, baik disebabkan karena dokter menolak penugasan, maupun sebab lainnya.
9	Menangani daftar penugasan dokter yang sudah dijadwalkan	Menampilkan daftar penugasan dokter pada dokter yang ditugaskan
10	Menangani konfirmasi	Melakukan proses konfirmasi

	penugasan oleh dokter yang ditugaskan	penugasan yang dilakukan oleh dokter, baik konfirmasi berupa persetujuan penugasan, penolakan penugasan, maupun konfirmasi penugasan selesai

3.1.3.2. Kebutuhan Non-Fungsional

Kebutuhan non-fungsional yang harus dipenuhi oleh sistem sebagai berikut:

1. Kebutuhan Performa
   Perangkat lunak akan berjalan dengan performa terbaik jika dijalankan di atas spesifikasi minimal.
2. Kebutuhan Perlindungan Keamanaan
   Username dan *password* dibutuhkan untuk mengamankan data pengguna yang berhak mengakses sistem sebagai syarat memasuki sistem dan melakukan semua fungsionalitas pada sistem.
3. Kualitas perangkat lunak dapat dilihat pada Tabel 3.2.

Tabel 3.2 Kualitas Perangkat Lunak

No	Parameter	Deskripsi
1	Ketersediaan	Aplikasi harus dapat berjalan pada sistem operasi yang sesuai dengan platform perangkat bergerak sesuai dengan platform yang telah disebutkan. Aplikasi dapat berjalan tanpa dibatasi waktu
2	Tingkat kualitas	Aplikasi dibangun dengan antarmuka pengguna yang konsisten, mudah dipahami dan mudah dioperasikan
3	*Portability*	Aplikasi mudah untuk dioperasikan pada *smartphone* dengan *platform* Android

3.1.3.3. Aktor

Pengertian pengguna adalah pihak-pihak, baik manusia maupun sistem atau perangkat lain yang terlibat dan berinteraksi

secara langsung dengan sistem. Pada perangkat lunak ini terdapat dua pengguna yaitu admin dan dokter. Admin adalah seseorang yang mengelola data pada sistem dan mengatur penjadwalan dokter untuk penanganan kondisi darurat. Sedangkan dokter adalah tenaga medis yang melakukan penanganan kondisi darurat di daerah bencana.

3.1.3.4. Kasus Penggunaan

Berdasarkan analisis spesifikasi kebutuhan fungsional dan analisis aktor dari sistem, dibuat kasus penggunaan sistem. Kasus penggunaan digambarkan dalam tabel penjelasan kasus penggunaan dan diagram kasus penggunaan. Tabel penjelasan kasus penggunaan dapat dilihat pada Tabel 3.3 dan diagram kasus penggunaan dapat dilihat pada Gambar 3.2.

Tabel 3.3 Kasus Penggunaan

Kode Kasus Penggunaan	Nama	Aktor
UC-0001	Melihat data daerah	Admin
UC-0002	Menambah data daerah	Admin
UC-0003	Mengedit data daerah	Admin
UC-0004	Menghapus data daerah	Admin
UC-0005	Melihat data bencana	Admin
UC-0006	Menambah data bencana	Admin
UC-0007	Mengedit data bencana	Admin
UC-0008	Menghapus data bencana	Admin
UC-0009	Melihat data dokter	Admin
UC-0010	Menambah data dokter	Admin
UC-0011	Mengedit data dokter	Admin
UC-0012	Menghapus data dokter	Admin
UC-0013	Melihat data rumah sakit	Admin

UC-0014	Menambah data rumah sakit	Admin
UC-0015	Mengedit data rumah sakit	Admin
UC-0016	Menghapus data rumah sakit	Admin
UC-0017	Melihat data jarak penugasan	Admin
UC-0018	Menambah data jarak penugasan	Admin
UC-0019	Mengedit data jarak penugasan	Admin
UC-0020	Melihat data keahlian dokter	Admin
UC-0021	Menambah data keahlian dokter	Admin
UC-0022	Mengedit data keahlian dokter	Admin
UC-0023	Menghapus data keahlian dokter	Admin
UC-0024	Mengatur penugasan dokter	Admin
UC-0025	Mengatur penugasan ulang dokter	Admin
UC-0026	Melihat daftar penugasan	Dokter
UC-0027	Menyetujui penugasan	Dokter
UC-0028	Menolak penugasan	Dokter
UC-0029	Memberikan konfirmasi penugasan selesai	Dokter

3.1.3.4.1 Melihat Data Daerah (UC-0001)

Kasus penggunaan nomor UC-0001 ini diakses ketika admin hendak melihat data daerah yang ada pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.4, Gambar 0.1, dan Gambar 0.30.

Tabel 3.4 Spesifikasi Kasus Penggunaan UC-0001

Kode Use Case	UC-0001	
Nama Use Case	Melihat data daerah	
Aktor	Admin	
Deskripsi	Admin dapat melihat data daerah	
Relasi	-	
Kondisi Awal	Sistem belum menampilkan data daerah	
Kondisi Akhir	Sistem sudah menampilkan data daerah	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Kelola Data"	
		2. Menampilkan halaman aktivitas "Kelola Data"
	3. Memilih pilihan "Kelola Data Daerah"	
		4. Menampilkan data daerah
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.2 Menambah Data Daerah (UC-0002)

Kasus penggunaan nomor UC-0002 ini diakses ketika admin hendak menambah data daerah pada sistem. Untuk menambah daerah, sistem akan menampilkan form input nama daerah dan rawan bencana pada daerah tersebut. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.5, Gambar 0.2, dan Gambar 0.31.

Tabel 3.5 Spesifikasi Kasus Penggunaan UC-0002

Kode Use Case	UC-0002
Nama Use Case	Menambah data daerah
Aktor	Admin

Deskripsi	Admin dapat menambah data daerah pada sistem	
Relasi	-	
Kondisi Awal	Admin belum menambahkan data daerah yang baru	
Kondisi Akhir	Sistem sudah menyimpan data daerah yang ditambah oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Tambah Daerah"	
		2. Menampilkan form yang berisi input "Nama Daerah" dan "Pilih Bencana"
	3. Mengisi "Nama Daerah"	
	4. Memilih "Pilih Bencana"	
		5.Menampilkan daftar rawan bencana
	6.Memilih satu atau lebih rawan bencana	
	7.a.Memilih "OK"	
		8.Menampilkan rawan bencana yang sudah dipilih
	9.Menekan tombol "Simpan"	
		10.Menyimpan data daerah yang baru ditambah
		11.Menampilkan semua data daerah pada halaman "Kelola Data Daerah"
Alur kejadian alternative	**Aktor**	**Sistem**
	7.b.Memilih "No"	

		7.b.1.Tidak menampilkan rawan bencana yang dipilih
	7.c.Memilih "Tambah"	
		7.c.1.Menampilkan form "Tambah Bencana"

3.1.3.4.3 Mengedit Data Daerah (UC-0003)

Kasus penggunaan nomor UC-0003 ini diakses ketika admin hendak merubah suatu data daerah. Pada kasus penggunaan ini, sistem menampilkan form untuk pengisian perubahan data pada daerah yang ingin dirubah. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.6, Gambar 0.3, dan Gambar 0.32.

Tabel 3.6 Spesifikasi Kasus Penggunaan UC-0003

Kode Use Case	UC-0003	
Nama Use Case	Mengedit data daerah	
Aktor	Admin	
Deskripsi	Admin dapat mengedit data daerah yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum mengedit data daerah yang perlu diedit	
Kondisi Akhir	Sistem sudah menyimpan data daerah yang diedit	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Edit Data Daerah"	
		2. Menampilkan form yang berisi "Nama Daerah" dan rawan

		bencana yang sudah ada sebelumnya
	3.Mengedit nama daerah pada bagian "Nama Daerah"	
	4.Memilih "Pilih Bencana"	
		5.Menampilkan daftar rawan bencana sesuai data yang ada sebelumnya
	6.Memilih rawan bencana yang dimaksud	
	7.a.Memilih "OK"	
		8.Menampilkan rawan bencana yang sudah diedit
	9.Menekan tombol "Simpan"	
		10.Menyimpan data daerah yang baru diedit
		11.Menampilkan semua data daerah pada halaman "Kelola Data Daerah"
Alur kejadian alternative	**Aktor**	**Sistem**
	7.b.Memilih "No"	
		7.b.1.Tidak menampilkan rawan bencana yang baru diedit
	7.c.Memilih"Tambah"	
		7.c.1.Menampilkan form "Tambah Bencana"

3.1.3.4.4 Menghapus Data Daerah (UC-0004)

Kasus penggunaan nomor UC-0004 ini dilakukan admin ketika akan menghapus suatu data daerah. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.7, Gambar 0.4, dan Gambar 0.33.

Tabel 3.7 Spesifikasi Kasus Penggunaan UC-0004

Kode Use Case	UC-0004	
Nama Use Case	Menghapus data daerah	
Aktor	Admin	
Deskripsi	Admin dapat menghapus data daerah yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum menghapus data daerah yang dimaksud	
Kondisi Akhir	Sistem sudah menghapus data daerah yang dihapus oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Hapus Data Daerah"	
		2. Menghapus data daerah yang sudah dipilih pada sistem
		3.Menampilkan semua data daerah pada halaman "Kelola Data Daerah"
Alur kejadian alternative	**Aktor**	**Sistem**

3.1.3.4.5 Melihat Data Bencana (UC-0005)

Kasus penggunaan nomor UC-0005 ini diakses ketika admin hendak melihat data bencana yang ada pada sistem.

Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.8, Gambar 0.5, dan Gambar 0.34.

Tabel 3.8 Spesifikasi Kasus Penggunaan UC-0005

Kode Use Case	UC-0005	
Nama Use Case	Melihat data bencana	
Aktor	Admin	
Deskripsi	Admin dapat melihat data bencana	
Relasi	-	
Kondisi Awal	Sistem belum menampilkan data bencana	
Kondisi Akhir	Sistem sudah menampilkan data bencana	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Kelola Data"	
		2. Menampilkan halaman aktivitas "Kelola Data"
	3. Memilih pilihan "Kelola Data Bencana"	
		4. Menampilkan data bencana
Alur kejadian alternative	**Aktor**	**Sistem**

3.1.3.4.6 Menambah Data Bencana (UC-0006)

Kasus penggunaan nomor UC-0006 ini diakses ketika admin hendak menambah data bencana pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.9, Gambar 0.6, dan Gambar 0.35.

Tabel 3.9 Spesifikasi Kasus Penggunaan UC-0006

Kode Use Case	UC-0006

Nama Use Case	Menambah data bencana	
Aktor	Admin	
Deskripsi	Admin dapat menambah data bencana pada sistem	
Relasi	-	
Kondisi Awal	Admin belum menambahkan data bencana yang baru	
Kondisi Akhir	Sistem sudah menyimpan data bencana yang ditambah oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Tambah Bencana"	
		2. Menampilkan form yang berisi input "Nama Bencana" dan "Pilih Keahlian"
	3. Mengisi "Nama Bencana"	
	4. Memilih "Pilih Keahlian"	
		5.Menampilkan daftar keahlian yang dibutuhkan
	6.Memilih satu atau lebih keahlian yang dibutuhkan	
	7.a.Memilih "OK"	
		8.Menampilkan keahlian yang sudah dipilih
	9.Menekan tombol Simpan	
		10.Menyimpan data bencana yang baru ditambah
		11.Menampilkan semua data daerah pada halaman "Kelola

		Data Bencana”
Alur kejadian alternatif	**Aktor**	**Sistem**
	7.b.Memilih “No”	
		7.b.1.Tidak menampilkan keahlian yang dipilih
	7.c.Memilih “Tambah”	
		7.c.1.Menampilkan form “Tambah Keahlian”

3.1.3.4.7 Mengedit Data Bencana (UC-0007)

Kasus penggunaan nomor UC-0007 ini diakses ketika admin hendak merubah suatu data bencana. Pada kasus penggunaan ini, sistem menampilkan form untuk pengisian perubahan data pada bencana yang ingin dirubah. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.10, Gambar 0.7, dan Gambar 0.36.

Tabel 3.10 Spesifikasi Kasus Penggunaan UC-0007

Kode Use Case	UC-0007	
Nama Use Case	Mengedit data bencana	
Aktor	Admin	
Deskripsi	Admin dapat mengedit data bencana yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum mengedit data bencana yang perlu diedit	
Kondisi Akhir	Sistem sudah menyimpan data bencana yang diedit	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan “Edit Bencana”	
		2. Menampilkan form yang berisi “Nama

		Bencana" dan keahlian yang sudah ada sebelumnya
	3.Mengedit nama bencana pada bagian "Nama Bencana" dalam isian yang sudah disediakan	
	4.Memilih "Pilih Keahlian"	
		5.Menampilkan daftar keahlian sesuai data yang ada sebelumnya
	6.Memilih keahlian yang dimaksud	
	7.a.Memilih "OK"	
		8.Menampilkan keahlian yang sudah diedit
	9.Menekan tombol "Simpan"	
		10.Menyimpan data bencana yang baru diedit
		11.Menampilkan semua data bencana pada halaman "Kelola Data Bencana"
Alur kejadian alternatif	**Aktor**	**Sistem**
	7.b.Memilih "No"	
		7.b.1.Tidak menampilkan keahlian yang baru diedit
	7.c.Memilih "Tambah"	
		7.c.1.Menampilkan form "Tambah

		Keahlian”

3.1.3.4.8 Menghapus Data Bencana (UC-0008)

Kasus penggunaan nomor UC-0008 ini diakses ketika ketika admin akan menghapus suatu data bencana. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.11, Gambar 0.8, dan Gambar 0.37.

Tabel 3.11 Spesifikasi Kasus Penggunaan UC-0008

Kode Use Case	UC-0008	
Nama Use Case	Menghapus data bencana	
Aktor	Admin	
Deskripsi	Admin dapat menghapus data bencana yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum menghapus data bencana yang dimaksud	
Kondisi Akhir	Sistem sudah menghapus data bencana yang dihapus oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan “Hapus Bencana”	
		2. Menghapus data bencana yang sudah dipilih pada sistem
		3.Menampilkan semua data bencana pada halaman “Kelola Data Bencana”
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.9 Melihat Data Dokter (UC-0009)

Kasus penggunaan nomor UC-0009 ini diakses ketika admin hendak melihat data bencana yang ada pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.12, Gambar 0.9, dan Gambar 0.38.

Tabel 3.12 Spesifikasi Kasus Penggunaan UC-0009

Kode Use Case	UC-0009	
Nama Use Case	Melihat data dokter	
Aktor	Admin	
Deskripsi	Admin dapat melihat data dokter	
Relasi	-	
Kondisi Awal	Sistem belum menampilkan data dokter	
Kondisi Akhir	Sistem sudah menampilkan data dokter	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Kelola Data"	
		2. Menampilkan halaman aktivitas "Kelola Data"
	3. Memilih pilihan "Kelola Data Dokter"	
		4. Menampilkan data dokter
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.10 Menambah Data Dokter (UC-0010)

Kasus penggunaan nomor UC-0010 ini diakses ketika admin hendak menambah data dokter pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.13, Gambar 0.10, dan Gambar 0.39.

Tabel 3.13 Spesifikasi Kasus Penggunaan UC-0010

Kode Use Case	UC-0010	
Nama Use Case	Menambah data dokter	
Aktor	Admin	
Deskripsi	Admin dapat menambah data dokter pada sistem	
Relasi	-	
Kondisi Awal	Admin belum menambahkan data dokter yang baru	
Kondisi Akhir	Sistem sudah menyimpan data dokter yang ditambah oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan “Tambah Dokter”	
		2. Menampilkan form yang berisi input “Nama Dokter” , “Pilih Keahlian”, dan “Pilih RS”
	3. Mengisi “Nama Dokter”	
	4. Memilih “Pilih Keahlian”	
		5.Menampilkan daftar keahlian
	6.Memilih satu atau lebih keahlian yang dimiliki dokter	
	7.a.Memilih “OK”	
		8.Menampilkan keahlian yang sudah dipilih
	9. Memilih “Pilih RS”	
		10.Menampilkan daftar rumah sakit

	11.Memilih salah satu rumah sakit tempat dokter bekerja	
	12.a.Memilih “OK”	
		13.Menampilkan rumah sakit yang sudah dipilih
	14.Menekan tombol Simpan	
		15.Menyimpan data dokter yang baru ditambah
		16.Menampilkan semua data dokter pada halaman “Kelola Data Dokter”
Alur kejadian alternatif	**Aktor**	**Sistem**
	7.b.Memilih “No”	
		7.b.1.Tidak menampilkan keahlian yang dipilih
	7.c.Memilih “Tambah”	
		7.c.1.Menampilkan form “Tambah Keahlian”
	12.b.Memilih “No”	
		12.b.1.Tidak menampilkan rumah sakit yang dipilih
	12.c.Memilih “Tambah”	
		12.c.1.Menampilkan form “Tambah RS”

3.1.3.4.11 Mengedit Data Dokter (UC-0011)

Kasus penggunaan nomor UC-0011 ini diakses ketika admin hendak merubah suatu data dokter. Pada kasus penggunaan ini, sistem menampilkan form untuk pengisian perubahan data pada dokter yang ingin dirubah. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.14, Gambar 0.11, dan Gambar 0.40.

Tabel 3.14 Spesifikasi Kasus Penggunaan UC-0011

Kode Use Case	UC-0011	
Nama Use Case	Mengedit data dokter	
Aktor	Admin	
Deskripsi	Admin dapat mengedit data dokter yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum mengedit data dokter yang perlu diedit	
Kondisi Akhir	Sistem sudah menyimpan data dokter yang diedit	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Edit Dokter"	
		2. Menampilkan form yang berisi "Nama Dokter", keahlian, dan rumah sakit yang sudah ada sebelumnya
	3.Mengedit nama dokter pada bagian "Nama Dokter" dalam isian yang sudah disediakan	
	4.Memilih "Pilih Keahlian"	
		5.Menampilkan daftar keahlian sesuai data

		yang ada sebelumnya
	6.Memilih keahlian yang dimaksud	
	7.a.Memilih “OK”	
		8.Menampilkan keahlian yang sudah diedit
	9.Memilih “Pilih RS”	
		10.Menampilkan rumah sakit sesuai data yang ada sebelumnya
	11.Memilih rumah sakit yang dimaksud	
	12.a.Memilih “OK”	
		13.Menampilkan rumah sakit yang sudah diedit
	14.Menekan tombol “Simpan”	
		15.Menyimpan data dokter yang baru diedit
		16.Menampilkan semua data dokter pada halaman “Kelola Data Dokter”
Alur kejadian alternatif	**Aktor**	**Sistem**
	7.b.Memilih “No”	
		7.b.1.Tidak menampilkan keahlian yang baru diedit
	7.c.Memilih “Tambah”	
		7.c.1.Menampilkan form “Tambah Keahlian”
	12.b.Memilih “No”	

		12.b.1.Tidak menampilkan rumah sakit yang dipilih
	12.c.Memilih "Tambah"	
		12.c.1.Menampilkan form "Tambah RS"

3.1.3.4.12 Menghapus Data Dokter (UC-0012)

Kasus penggunaan nomor UC-0012 ini diakses ketika ketika admin akan menghapus suatu data dokter. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.15, Gambar 0.12, dan Gambar 0.41.

Tabel 3.15 Spesifikasi Kasus Penggunaan UC-0012

Kode Use Case	UC-0012	
Nama Use Case	Menghapus data dokter	
Aktor	Admin	
Deskripsi	Admin dapat menghapus data dokter yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum menghapus data dokter yang dimaksud	
Kondisi Akhir	Sistem sudah menghapus data dokter yang dihapus oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Hapus Dokter"	
		2. Menghapus data dokter yang sudah dipilih pada sistem
		3.Menampilkan semua data dokter pada halaman "Kelola Data Dokter"

Alur kejadian alternative	Aktor	Sistem

3.1.3.4.13 Melihat Data Rumah Sakit (UC-0013)

Kasus penggunaan nomor UC-0013 ini diakses ketika admin hendak melihat data rumah sakit yang ada pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.16, Gambar 0.13, dan Gambar 0.42.

Tabel 3.16 Spesifikasi Kasus Penggunaan UC-0013

Kode Use Case	UC-0013	
Nama Use Case	Melihat data rumah sakit	
Aktor	Admin	
Deskripsi	Admin dapat melihat data rumah sakit	
Relasi	-	
Kondisi Awal	Sistem belum menampilkan data rumah sakit	
Kondisi Akhir	Sistem sudah menampilkan data rumah sakit	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Kelola Data"	
		2. Menampilkan halaman aktivitas "Kelola Data"
	3. Memilih pilihan "Kelola Data Rumah Sakit"	
		4. Menampilkan data rumah sakit
Alur kejadian alternative	**Aktor**	**Sistem**

3.1.3.4.14 Menambah Data Rumah Sakit (UC-0014)

Kasus penggunaan nomor UC-0014 ini diakses ketika admin hendak menambah data rumah sakit pada sistem.

Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.17, Gambar 0.14, dan Gambar 0.43.

Tabel 3.17 Spesifikasi Kasus Penggunaan UC-0014

Kode Use Case	UC-0014	
Nama Use Case	Menambah data bencana	
Aktor	Admin	
Deskripsi	Admin dapat menambah data rumah sakit pada sistem	
Relasi	-	
Kondisi Awal	Admin belum menambahkan data rumah sakit yang baru	
Kondisi Akhir	Sistem sudah menyimpan data rumah sakit yang ditambah oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Tambah RS"	
		2. Menampilkan form yang berisi input "Nama RS" dan "Alamat RS"
	3. Mengisi "Nama RS"	
	4. Mengisi "Alamat RS"	
	5.Menekan tombol "Simpan"	
		6.Menyimpan data rumah sakit yang baru ditambah
		7.Menampilkan semua data rumah sakit pada halaman "Kelola Data RS"
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.15 Mengedit Data Rumah Sakit (UC-0015)

Kasus penggunaan nomor UC-0015 ini diakses ketika admin hendak merubah suatu data rumah sakit. Pada kasus penggunaan ini, sistem menampilkan form untuk pengisian perubahan data pada rumah sakit yang ingin dirubah. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.18, Gambar 0.15, dan Gambar 0.44.

Tabel 3.18 Spesifikasi Kasus Penggunaan UC-0015

Kode Use Case	UC-0015	
Nama Use Case	Mengedit data rumah sakit	
Aktor	Admin	
Deskripsi	Admin dapat mengedit data rumah sakit yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum mengedit data rumah sakit yang perlu diedit	
Kondisi Akhir	Sistem sudah menyimpan data rumah sakit yang diedit	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Edit RS"	
		2. Menampilkan form yang berisi "Nama RS" dan "Alamat RS" yang sudah ada sebelumnya
	3.Mengedit nama rumah sakit pada bagian "Nama RS" dalam isian yang sudah disediakan	
	4. Mengedit alamat rumah sakit pada bagian "Alamat RS" dalam isian	

	yang sudah disediakan	
	5.Menekan tombol "Simpan"	
		6.Menyimpan data rumah sakit yang baru diedit
		7.Menampilkan semua data rumah sakit pada halaman "Kelola Data RS"
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.16 Menghapus Data Rumah Sakit (UC-0016)

Kasus penggunaan nomor UC-0016 ini diakses ketika ketika admin akan menghapus suatu data rumah sakit. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.19, Gambar 0.16, dan Gambar 0.45.

Tabel 3.19 Spesifikasi Kasus Penggunaan UC-0016

Kode Use Case	UC-0016	
Nama Use Case	Menghapus data rumah sakit	
Aktor	Admin	
Deskripsi	Admin dapat menghapus data rumah sakit yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum menghapus data rumah sakit yang dimaksud	
Kondisi Akhir	Sistem sudah menghapus data rumah sakit yang dihapus oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Hapus RS"	

		2. Menghapus data rumah sakit yang sudah dipilih pada sistem
		3.Menampilkan semua data rumah sakit pada halaman "Kelola Data RS"
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.17 Melihat Data Jarak Penugasan (UC-0017)

Kasus penggunaan nomor UC-0017 ini diakses ketika admin hendak melihat data jarak penugasan yang ada pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.20, Gambar 0.17, dan Gambar 0.46.

Tabel 3.20 Spesifikasi Kasus Penggunaan UC-0017

Kode Use Case	UC-0017	
Nama Use Case	Melihat data jarak penugasan	
Aktor	Admin	
Deskripsi	Admin dapat melihat data jarak penugasan	
Relasi	-	
Kondisi Awal	Sistem belum menampilkan data jarak penugasan	
Kondisi Akhir	Sistem sudah menampilkan data jarak penugasan	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Kelola Data"	
		2. Menampilkan halaman aktivitas "Kelola Data"
	3. Memilih pilihan "Kelola Data Jarak"	
		4. Menampilkan data jarak penugasan
Alur kejadian	**Aktor**	**Sistem**

alternatif		

3.1.3.4.18 Menambah Data Jarak Penugasan (UC-0018)

Kasus penggunaan nomor UC-0018 ini diakses ketika admin hendak menambah data jarak penugasan pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.21, Gambar 0.18, dan Gambar 0.47.

Tabel 3.21 Spesifikasi Kasus Penggunaan UC-0018

Kode Use Case	UC-0018	
Nama Use Case	Menambah data jarak penugasan	
Aktor	Admin	
Deskripsi	Admin dapat menambah data jarak penugasan pada sistem	
Relasi	-	
Kondisi Awal	Admin belum menambahkan data jarak penugasan yang baru	
Kondisi Akhir	Sistem sudah menyimpan data jarak penugasan yang ditambah oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Tambah Jarak"	
		2. Menampilkan form yang berisi input "Pilih RS", "Pilih Daerah Bencana", dan "Jarak"
	3. Memilih "Pilih RS"	
		4.Menampilkan daftar rumah sakit
	5.Memilih salah satu rumah sakit yang akan disimpan data jarak	

	penugasannya	
	6.a.Memilih “OK”	
		7.Menampilkan rumah sakit yang sudah dipilih
	8. Memilih “Pilih Daerah Bencana”	
		9.Menampilkan daftar daerah
	10.Memilih salah satu daerah yang akan disimpan data jarak penugasannya	
	11.a.Memilih “OK”	
		12.Menampilkan daerah yang sudah dipilih
	13. Mengisi “Jarak”	
	14.Menekan tombol “Simpan”	
		15.Menyimpan data jarak penugasan yang baru ditambah
		16.Menampilkan semua data jarak penugasan pada halaman “Kelola Data Jarak”
Alur kejadian alternatif	**Aktor**	**Sistem**
	6.b.Memilih “No”	
		6.b.1.Tidak menampilkan rumah sakit yang dipilih
	6.c.Memilih “Tambah”	
		6.c.1.Menampilkan form “Tambah RS”
	11.b.Memilih “No”	

		11.b.1.Tidak menampilkan daerah yang dipilih
	11.c.Memilih "Tambah"	
		11.c.1.Menampilkan form "Tambah Daerah"

3.1.3.4.19 Mengedit Data Jarak Penugasan (UC-0019)

Kasus penggunaan nomor UC-0019 ini diakses ketika admin hendak merubah besar jarak pada data jarak penugasan. Pada kasus penggunaan ini, sistem menampilkan form untuk pengisian perubahan data jarak pada jarak penugasan yang ingin dirubah. Spesifikasi, diagram aktivitas, dan sekuens kolaborasi kasus penggunaan ini dapat dilihat pada Tabel 3.22, Gambar 0.19, dan Gambar 0.48.

Tabel 3.22 Spesifikasi Kasus Penggunaan UC-0019

Kode Use Case	UC-0019	
Nama Use Case	Mengedit data jarak penugasan	
Aktor	Admin	
Deskripsi	Admin dapat mengedit data jarak penugasan yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum mengedit data jarak penugasan yang perlu diedit	
Kondisi Akhir	Sistem sudah menyimpan data jarak penugasan yang diedit	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Edit Jarak"	
		2. Menampilkan form yang berisi "RS yang

		dipilih", "Daerah Bencana", dan "Jarak" yang sudah ada sebelumnya
	3.Mengedit jarak penugasan pada bagian "Jarak" dalam isian yang sudah disediakan	
	4.Menekan tombol "Simpan"	
		5.Menyimpan data jarak penugasan yang baru diedit
		6.Menampilkan semua data jarak penugasan pada halaman "Kelola Data Jarak"
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.20 Melihat Data Keahlian Dokter (UC-0020)

Kasus penggunaan nomor UC-0020 ini diakses ketika admin hendak melihat data keahlian dokter yang ada pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.23, Gambar 0.20, dan Gambar 0.49.

Tabel 3.23 Spesifikasi Kasus Penggunaan UC-0020

Kode Use Case	UC-0020
Nama Use Case	Melihat data keahlian dokter
Aktor	Admin
Deskripsi	Admin dapat melihat data keahlian dokter
Relasi	-
Kondisi Awal	Sistem belum menampilkan data keahlian dokter

Kondisi Akhir	Sistem sudah menampilkan data keahlian dokter	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Kelola Data"	
		2. Menampilkan halaman aktivitas "Kelola Data"
	3. Memilih pilihan "Kelola Data Keahlian"	
		4. Menampilkan data keahlian dokter
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.21 Menambah Data Keahlian Dokter (UC-0021)

Kasus penggunaan nomor UC-0021 ini diakses ketika admin hendak menambah data keahlian dokter pada sistem. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.24, Gambar 0.21, dan Gambar 0.50.

Tabel 3.24 Spesifikasi Kasus Penggunaan UC-0021

Kode Use Case	UC-0021	
Nama Use Case	Menambah data keahlian dokter	
Aktor	Admin	
Deskripsi	Admin dapat menambah data keahlian dokter pada sistem	
Relasi	-	
Kondisi Awal	Admin belum menambahkan data keahlian dokter yang baru	
Kondisi Akhir	Sistem sudah menyimpan data keahlian dokter yang ditambah oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan	

	“Tambah Keahlian”	
		2. Menampilkan form yang berisi input “Nama Keahlian”
	3. Mengisi “Nama Keahlian”	
	4.Menekan tombol Simpan	
		5.Menyimpan data keahlian dokter yang baru ditambah
		6.Menampilkan semua data keahlian dokter pada halaman “Kelola Data Keahlian”
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.22 Mengedit Data Keahlian Dokter (UC-0022)

Kasus penggunaan nomor UC-0022 ini diakses ketika admin hendak merubah suatu data keahlian dokter. Pada kasus penggunaan ini, sistem menampilkan form untuk pengisian perubahan data pada keahlian dokter yang ingin dirubah. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.25, Gambar 0.22, dan Gambar 0.51.

Tabel 3.25 Spesifikasi Kasus Penggunaan UC-0022

Kode Use Case	UC-0022
Nama Use Case	Mengedit data keahlian dokter
Aktor	Admin
Deskripsi	Admin dapat mengedit data keahlian dokter yang diperlukan
Relasi	-
Kondisi Awal	Admin belum mengedit data keahlian dokter

	yang perlu diedit	
Kondisi Akhir	Sistem sudah menyimpan data keahlian dokter yang diedit	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Edit Keahlian"	
		2. Menampilkan form yang berisi "Nama Keahlian" yang sudah ada sebelumnya
	3.Mengedit nama keahlian dokter pada bagian "Nama Keahlian" dalam isian yang sudah disediakan	
	4.Menekan tombol "Simpan"	
		5.Menyimpan data keahlian dokter yang baru diedit
		6.Menampilkan semua data keahlian dokter pada halaman "Kelola Data Keahlian"
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.23 Menghapus Data Keahlian Dokter (UC-0023)

Kasus penggunaan nomor UC-0023 ini diakses ketika ketika admin akan menghapus suatu data keahlian dokter. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.26, Gambar 0.23, dan Gambar 0.52.

Tabel 3.26 Spesifikasi Kasus Penggunaan UC-0023

Kode Use Case	UC-0023	
Nama Use Case	Menghapus data keahlian dokter	
Aktor	Admin	
Deskripsi	Admin dapat menghapus data keahlian dokter yang diperlukan	
Relasi	-	
Kondisi Awal	Admin belum menghapus data keahlian dokter yang dimaksud	
Kondisi Akhir	Sistem sudah menghapus data keahlian dokter yang dihapus oleh admin	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Hapus Keahlian"	
		2. Menghapus data keahlian dokter yang sudah dipilih pada sistem
		3.Menampilkan semua data keahlian dokter pada halaman "Kelola Data Keahlian"
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.24 Mengatur Penugasan Dokter (UC-0024)

Kasus penggunaan nomor UC-0024 ini diakses ketika admin hendak menjadwalkan penugasan dokter ke daerah yang memiliki kondisi darurat. Dalam penugasan ini, setiap daerah bencana tepat ditangani oleh 1 dokter. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.27, Gambar 0.24, dan Gambar 0.53.

Tabel 3.27 Spesifikasi Kasus Penggunaan UC-0024

Kode Use Case	UC-0024	
Nama Use Case	Mengatur penugasan dokter	
Aktor	Admin	
Deskripsi	Admin dapat memperoleh hasil penugasan dokter yang optimal ke daerah-daerah bencana yang ada	
Relasi	-	
Kondisi Awal	Admin belum mengatur jadwal penugasan dokter ke daerah bencana	
Kondisi Akhir	Sistem menampilkan dan menyimpan hasil penugasan optimal, berupa jarak penugasan dan dokter mana saja yang ditugaskan pada masing-masing daerah bencana	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan "Penugasan"	
		2.Menampilkan halaman "Penugasan Dokter"
	3. Memilih "Pilih Dokter"	
		4.Menampilkan daftar dokter
	5.Memilih satu atau lebih dokter yang hendak ditugaskan	
	6.a.Memilih "OK"	
		7.Menampilkan dokter yang sudah dipilih
	8. Memilih "Pilih Daerah"	
		9.Menampilkan daftar daerah
	10.Memilih satu atau lebih daerah yang perlu	

	penanganan darurat	
	11.a.Memilih “OK”	
		12.Menampilkan daerah yang sudah dipilih
	13.Menekan tombol “Submit”	
		14. Menampilkan total jarak penugasan, dokter yang ditugaskan pada tiap daerah bencana, beserta masing-masing jarak penugasan dokter ke daerah bencana tersebut
	15. Menekan tombol “Simpan”	
		16.Menyimpan data penugasan
Alur kejadian alternatif	**Aktor**	**Sistem**
	6.b.Memilih “No”	
		6.b.1.Tidak menampilkan dokter yang dipilih
	6.c.Memilih “Pilih semua”	
		6.c.1.Memilih semua dokter dan menampilkan semua dokter yang dipilih
	11.b.Memilih “No”	
		11.b.1.Tidak menampilkan daerah yang dipilih

	11.c.Memilih “Pilih semua”	
		11.c.1.Memilih semua daerah dan menampilkan semua daerah yang dipilih

3.1.3.4.25 Mengatur Penugasan Ulang Dokter (UC-0025)

Kasus penggunaan nomor UC-0025 ini diakses ketika admin hendak menjadwalkan ulang penugasan dokter ke daerah darurat dengan memperoleh dokter pengganti. Pada kasus penggunaan ini, daftar dokter pengganti yang ditampilkan sebagai masukan pada sistem adalah dokter yang tidak sedang melaksanakan penugasan. Hasil dari proses ini adalah dokter dengan jarak penugasan paling optimal untuk penugasan ke daerah yang dituju. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.28, Gambar 0.25, dan Gambar 0.54.

Tabel 3.28 Spesifikasi Kasus Penggunaan UC-0025

Kode Use Case	UC-0025
Nama Use Case	Mengatur penugasan ulang dokter
Aktor	Admin
Deskripsi	Admin dapat mencari dokter pengganti untuk penugasan ke daerah bencana dan memperoleh hasil penugasan ke daerah tersebut berupa dokter pengganti dan jarak penugasan yang optimal
Relasi	-
Kondisi Awal	Admin belum memperoleh dokter pengganti yang ditugaskan ke daerah yang perlu untuk dijadwal ulang
Kondisi Akhir	Sistem menampilkan dan menyimpan hasil penugasan optimal, berupa dokter pengganti dan jarak penugasan ke daerah bencana yang

	dijadwal ulang	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih “Cari Dokter Pengganti”	
		2.Menampilkan halaman “Penugasan Ulang Dokter”
	3. Memilih “Pilih Dokter”	
		4.Menampilkan daftar dokter pengganti
	5.Memilih satu atau lebih dokter yang perlu ditugaskan	
	6.a.Memilih “OK”	
		7.Menampilkan dokter yang sudah dipilih
	8.Menekan tombol “Submit”	
		9. Menampilkan dokter pengganti dan jarak penugasan yang ditugaskan pada daerah bencana
	10. Menekan tombol “Simpan”	
		11.Menyimpan data pada penugasan yang diulang tersebut
Alur kejadian alternatif	**Aktor**	**Sistem**
	6.b.Memilih “No”	
		6.b.1.Tidak menampilkan dokter yang dipilih
	6.c.Memilih “Pilih semua”	

		6.c.1.Memilih semua dokter dan menampilkan semua dokter yang dipilih

3.1.3.4.26 Melihat Daftar Penugasan (UC-0026)

Kasus penggunaan nomor UC-0026 ini diakses ketika dokter hendak melihat jadwal penugasan yang diajukan kepadanya dari admin. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.29, Gambar 0.26, dan Gambar 0.55.

Tabel 3.29 Spesifikasi Kasus Penggunaan UC-0026

Kode Use Case	UC-0026	
Nama Use Case	Melihat daftar penugasan	
Aktor	Dokter	
Deskripsi	Dokter dapat melihat daftar penugasan yang diajukan oleh admin	
Relasi	-	
Kondisi Awal	Sistem belum menampilkan daftar penugasan pada dokter yang ditugaskan	
Kondisi Akhir	Sistem sudah menampilkan daftar penugasan pada dokter yang ditugaskan	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan “Home Dokter”	
	2. Menekan tombol “Daftar Penugasan”	
		3. Menampilkan daftar penugasan yang diajukan kepada dokter yang mengakses sistem pada halaman “Daftar Penugasan”
Alur kejadian alternatif	**Aktor**	**Sistem**

3.1.3.4.27 Menyetujui Penugasan (UC-0027)

Kasus penggunaan nomor UC-0027 ini diakses ketika dokter hendak menyetujui penugasan yang diajukan oleh admin. Dokter diasumsikan dapat menyetujui penugasan lebih dari satu penugasan. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.30, Gambar 0.27, dan Gambar 0.56.

Tabel 3.30 Spesifikasi Kasus Penggunaan UC-0027

Kode Use Case	UC-0027	
Nama Use Case	Menyetujui penugasan	
Aktor	Dokter	
Deskripsi	Dokter menerima penugasan yang diajukan dengan cara menyetujui penugasan yang ada pada halaman "Daftar Penugasan"	
Relasi	-	
Kondisi Awal	Dokter belum menyetujui penugasan dari halaman "Daftar Penugasan"	
Kondisi Akhir	Dokter sudah menyetujui penugasan yang ada pada halaman "Daftar Penugasan"	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih satu penugasan yang ada pada halaman "Daftar Penugasan"	
		2. Menampilkan "Konfirmasi Penugasan" berupa pertanyaan "Apakah Anda mengambil penugasan ini?"
	3. a.Memilih "Iya"	
		4. Menyimpan penugasan
		5. Menampilkan konfirmasi penugasan

		berhasil dipilih
Alur kejadian alternatif	**Aktor**	**Sistem**
	3.b. Memilih "Kembali"	
		3.b.1 Kembali ke halaman "Daftar Penugasan"

3.1.3.4.28 Menolak Penugasan (UC-0028)

Kasus penggunaan nomor UC-0028 ini diakses ketika dokter hendak menolak penugasan yang diajukan oleh admin. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.31, Gambar 0.28, dan Gambar 0.57.

Tabel 3.31 Spesifikasi Kasus Penggunaan UC-0028

Kode Use Case	UC-0028	
Nama Use Case	Menolak penugasan	
Aktor	Dokter	
Deskripsi	Dokter dapat menolak penugasan yang ada pada halaman "Daftar Penugasan"	
Relasi	-	
Kondisi Awal	Dokter belum menolak penugasan dari halaman "Daftar Penugasan"	
Kondisi Akhir	Dokter sudah menolak penugasan yang ada pada halaman "Daftar Penugasan"	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih satu penugasan yang ada pada halaman "Daftar Penugasan"	
		2. Menampilkan "Konfirmasi Penugasan" berupa pertanyaan "Apakah Anda mengambil penugasan ini?"

	3. a.Memilih "Tidak"	
		4. Menampilkan halaman daftar penugasan dan tidak menampilkan lagi daftar penugasan yang sudah ditolak tersebut
Alur kejadian alternatif	**Aktor**	**Sistem**
	3.b. Memilih "Kembali"	
		3.b.1 Kembali ke halaman "Daftar Penugasan"

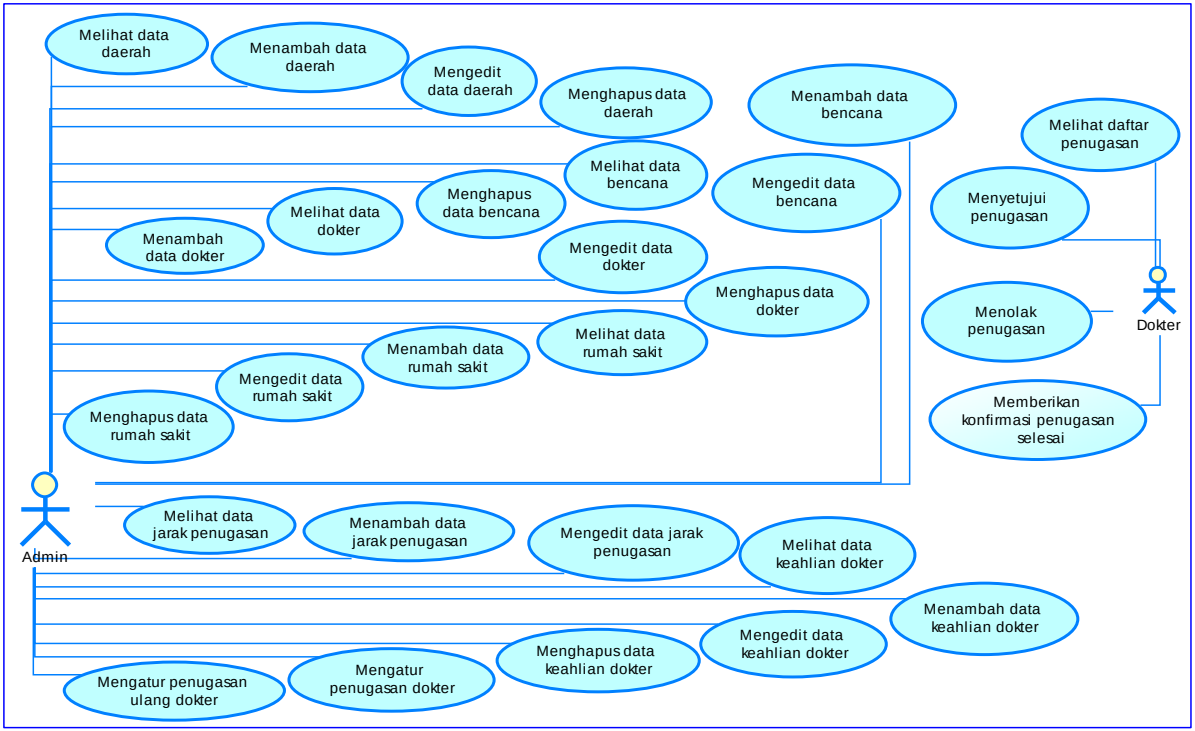
3.1.3.4.29 Memberikan Konfirmasi Penugasan Selesai (UC-0029)

Kasus penggunaan nomor UC-0029 ini diakses ketika dokter hendak memberikan konfirmasi penugasan selesai dilaksanakan. Spesifikasi, diagram aktivitas, dan sekuens kasus penggunaan ini dapat dilihat pada Tabel 3.32, Gambar 0.29, dan Gambar 0.58.

Tabel 3.32 Spesifikasi Kasus Penggunaan UC-0029

Kode Use Case	UC-0029	
Nama Use Case	Memberikan konfirmasi penugasan selesai	
Aktor	Dokter	
Deskripsi	Dokter memberi konfirrmasi setelah selesai melaksanakan penugasan	
Relasi	-	
Kondisi Awal	Dokter belum memberi konfirmasi penugasan selesai pada sistem	
Kondisi Akhir	Sistem sudah menyimpan konfirmasi penugasan selesai	
Alur kejadian normal	**Aktor**	**Sistem**
	1. Memilih pilihan	

	"Riwayat"	
		2. Menampilkan halaman "Riwayat"
	3. Memilih "Progres"	
		4. Menampilkan halaman "Progres"
	5. Memilih daftar penugasan yang sudah selesai dilaksanakan	
		6. Menampilkan detil penugasan dan "Konfirmasi Selesai Penugasan" berupa pertanyaan "Apakah Anda telah menyelesaikan Penugasan?" pada halaman aktivitas
	3. a.Memilih "Sudah"	
		4. Menyimpan riwayat penugasan selesai dan menghapus data penugasan tersebut dari halaman "Progres"
		5. Menampilkan halaman "Progres"
Alur kejadian alternative	**Aktor**	**Sistem**
	3.b. Memilih "Belum"	
		3.b.1 Kembali ke halaman "Progres"



Gambar 3.2 Diagram Kasus Penggunaan

3.2 Perancangan

Perancangan dalam subbab ini membahas perancangan dari aplikasi tugas akhir. Subbab ini terdiri dari lingkungan perancangan perangkat lunak, perancangan arsitektur sistem, perancangan diagram kelas, perancangan basis data, dan perancangan antarmuka pengguna.

3.2.1 Lingkungan Perancangan Perangkat Lunak

Spesifikasi perangkat keras serta perangkat lunak yang digunakan dalam tahap perancangan perangkat lunak tugas akhir ini seperti dijelaskan pada Tabel 3.33.

Tabel 3.33 Lingkungan Perancangan Perangkat Lunak

Perangkat Keras	Komputer	HP Pavilion Sleekbook 15
	Prosesor	Intel® Core™ i5-3230M CPU @ 2.60GHz (2.60GHz)
	Memori Primer	8 GB
	Memori Sekunder	500 GB
Perangkat Lunak	Sistem Operasi	*Windows* 8.1 Pro 64-bit
	Perangkat Lunak	Android Studio v2.2.2, Paint, CorelDraw X7, Sybase PowerDesigner 16.5, Microsoft Word 2013

3.2.2 Perancangan Arsitektur Sistem

Pada arsitektur sistem ini, *smartphone* digunakan untuk mengakses aplikasi *mobile* penugasan dokter. Proses bisnis yang ada pada aplikasi ditampilkan melalui antarmuka sistem.

Dalam proses aktivitas aplikasi, data yang digunakan diperoleh dan diminta melalui *web service*. Aplikasi akan mengirim 2 jenis *request*, antara lain *string request* dan JSON *object request*. Sistem mengirim *string request* ke *web service* pada kasus penggunaan UC-0002, UC-0003, UC-0004, UC-0006, UC-0007, UC-0008, UC-0010, UC-0011, UC-0012, UC-0014, UC-0015, UC-0016, UC-0018, UC-0019, UC-0021, UC-0022, UC-0023, UC-0024, UC-0025, UC-0026, UC-0027, UC-0028, dan UC-0029. Sistem pada *web service* akan menerima data dari aplikasi dan menyimpannya ke dalam *database* yang sudah dibuat. Jika data berhasil disimpan, *query response* memberikan nilai *true* dan *web service* akan mengirimkan JSON *post response* ke aplikasi.

Di sisi lain, sistem mengirim JSON *object request* ke *web service* pada kasus penggunaan UC-0001, UC-0002, UC-0003, UC-0005, UC-0006, UC-0007, UC-0009, UC-0010, UC-0011, UC-0013, UC-0017, UC-0018, UC-0020, dan UC-0024 yang merupakan kasus penggunaan yang memerlukan data dari *database* melalui *web service* berupa JSON *object*. Setelah JSON *object request* dikirim ke *web service*, sistem pada *web service* akan melakukan *request* pengambilan data ke *database*. Jika data berhasil diambil, *web service* akan mengirim JSON *get response* ke aplikasi. Perancangan arsitektur sistem pada aplikasi ini dapat dilihat pada Gambar 3.1.

3.2.3 Perancangan Diagram Kelas

Perancangan Diagram Kelas dapat dilihat pada Gambar 3.3, Gambar 3.4, Gambar 3.5, Gambar 3.6, Gambar 3.7, Gambar 3.8, Gambar 3.9, dan Gambar 3.10.

3.2.4 Perancangan Struktur Data

Dalam membuat suatu aplikasi perangkat bergerak, diperlukan analisis kebutuhan berupa perancangan basis data. Basis data yang digunakan kelak adalah MySQL. MySQL dipilih menjadi basis data aplikasi ini karena sifat RDBMS yang *open source*, mudah digunakan, dan memiliki *performance tuning*, yaitu menangani *query* sederhana dengan cepat [4].

Rancangan basis data ditampilkan dalam bentuk *Conceptual Data Model* (selanjutnya disebut CDM) dan *Physical Data Model* (selanjutnya disebut PDM). Penjelasan lebih lengkap berupa CDM dan PDM terdapat pada Gambar 3.11 dan Gambar 3.12.

3.2.4.1 Tabel Akun

Tabel akun adalah tabel yang digunakan untuk menyimpan data-data para pengguna yang memiliki hak akses pada sistem. Tabel ini merupakan tabel utama yang mempunyai atribut-atribut, yaitu id akun, *username*, *password*, dan hak akses. Sistem ini hanya menyediakan dua hak akses, yaitu administrator dan dokter.

3.2.4.2 Tabel Admin

Tabel admin adalah tabel yang digunakan untuk menyimpan data-data admin. Tabel ini merupakan tabel utama yang mempunyai atribut-atribut, yaitu id admin, id akun, nama admin, dan email admin.

3.2.4.3 Tabel Bencana

Tabel bencana adalah tabel yang digunakan untuk menyimpan data-data bencana. Tabel ini merupakan tabel utama yang mempunyai atribut-atribut, yaitu id bencana dan nama bencana.

3.2.4.4 Tabel Jarak

Tabel jarak adalah tabel yang digunakan untuk menyimpan jarak yang ada dalam perjalanan dari suatu rumah sakit ke daerah bencana. Rumah sakit diasumsikan sebagai lokasi dari dokter yang ditugaskan untuk menangani bencana pada suatu daerah. Tabel jarak ini merupakan tabel utama yang mempunyai atribut-atribut yaitu id jarak, id rumah sakit, id daerah, dan jarak.

3.2.4.5 Tabel Daerah

Tabel daerah adalah tabel yang digunakan untuk menyimpan data-data daerah. Tabel ini merupakan tabel utama yang mempunyai atribut-atribut, yaitu id daerah dan nama daerah.

3.2.4.6 Tabel Dokter

Tabel dokter adalah tabel yang digunakan untuk menyimpan data-data dokter. Tabel ini merupakan tabel utama yang mempunyai atribut-atribut, yaitu id dokter, id akun, id rumah sakit, nama dokter, email dokter, dan status dokter. Status dokter adalah atribut yang berguna untuk menandakan dokter yang sedang dalam penugasan atau tidak.

3.2.4.7 Tabel Keahlian Dokter

Tabel keahlian dokter adalah tabel yang digunakan untuk menyimpan data-data keahlian dokter. Tabel ini merupakan tabel utama yang mempunyai atribut-atribut, yaitu id keahlian dan nama keahlian.

3.2.4.8 Tabel Memerlukan

Tabel memerlukan adalah tabel yang menyimpan data-data yang berhubungan dengan tabel bencana dan tabel keahlian. Tabel ini merupakan tabel pendukung hasil dari relasi tabel bencana dan tabel keahlian. Atribut yang dimiliki pada tabel memerlukan adalah id bencana dan id keahlian.

3.2.4.9 Tabel Memiliki

Tabel memiliki adalah tabel yang menyimpan data-data yang berhubungan dengan tabel dokter dan tabel keahlian. Tabel ini merupakan tabel pendukung hasil dari relasi tabel dokter dan tabel keahlian. Atribut yang dimiliki pada tabel memiliki adalah id dokter dan id keahlian.

3.2.4.10 Tabel Penugasan

Tabel penugasan adalah tabel yang digunakan untuk menyimpan data-data hasil penugasan. Tabel ini merupakan tabel utama yang mempunyai atribut-atribut, yaitu id penugasan, waktu, dan status sebagai identifikasi suatu penugasan masih menunggu semua konfirmasi dokter, disetujui semua dokter yang ditugaskan, ditolak, atau semua dokter sudah selesai melaksanakan penugasan.

3.2.4.11 Tabel Penugasan Detil

Tabel penugasan detil adalah tabel yang digunakan untuk menyimpan data-data detil dari hasil penugasan. Tabel ini merupakan tabel pendukung yang mempunyai atribut-atribut, yaitu id penugasan, id daerah, id dokter, id admin, dan status detil. Data-data pada tabel ini berguna untuk menyimpan detil dari suatu penugasan.

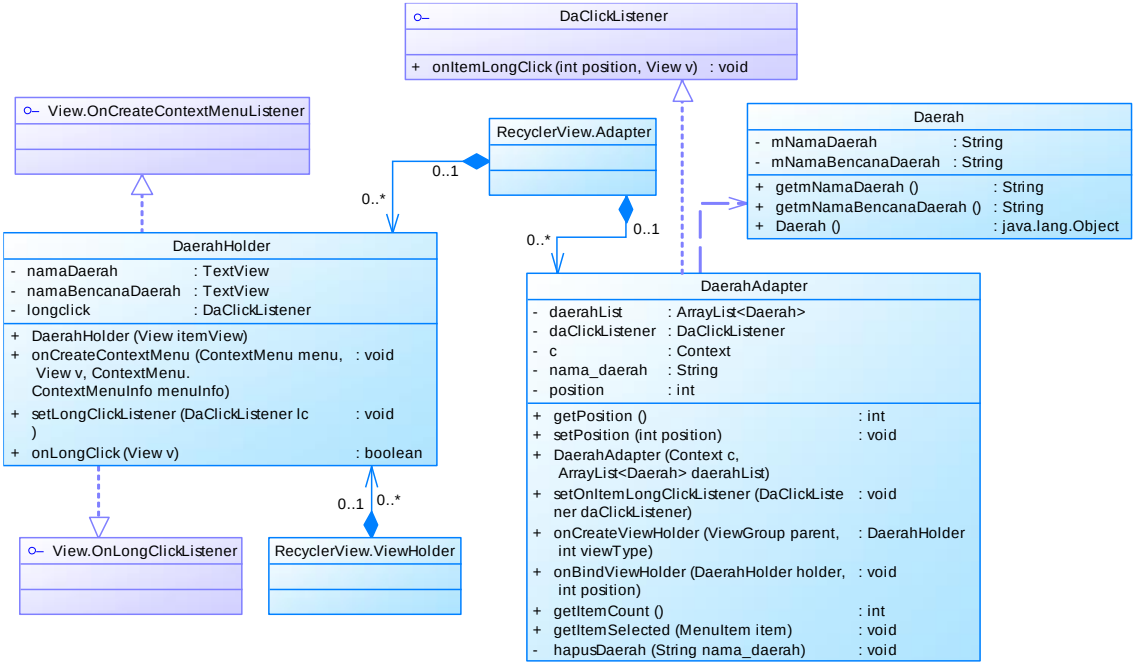
3.2.4.12 Tabel Rawan

Tabel rawan adalah tabel yang digunakan untuk menyimpan data-data yang berhubungan dengan tabel daerah dan tabel bencana. Tabel ini merupakan tabel pendukung hasil dari relasi tabel daerah dan tabel bencana. Atribut yang dimiliki pada tabel rawan adalah id daerah dan id bencana.

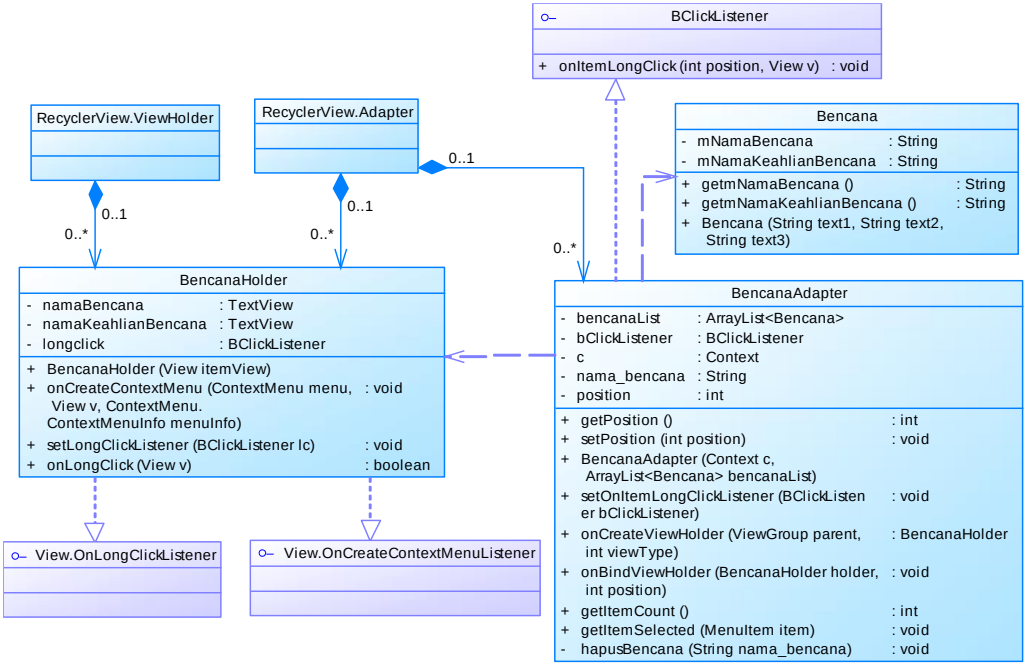
3.2.4.13 Tabel Rumah Sakit

Tabel rumah sakit adalah tabel yang digunakan untuk menyimpan data-data rumah sakit. Tabel ini merupakan tabel

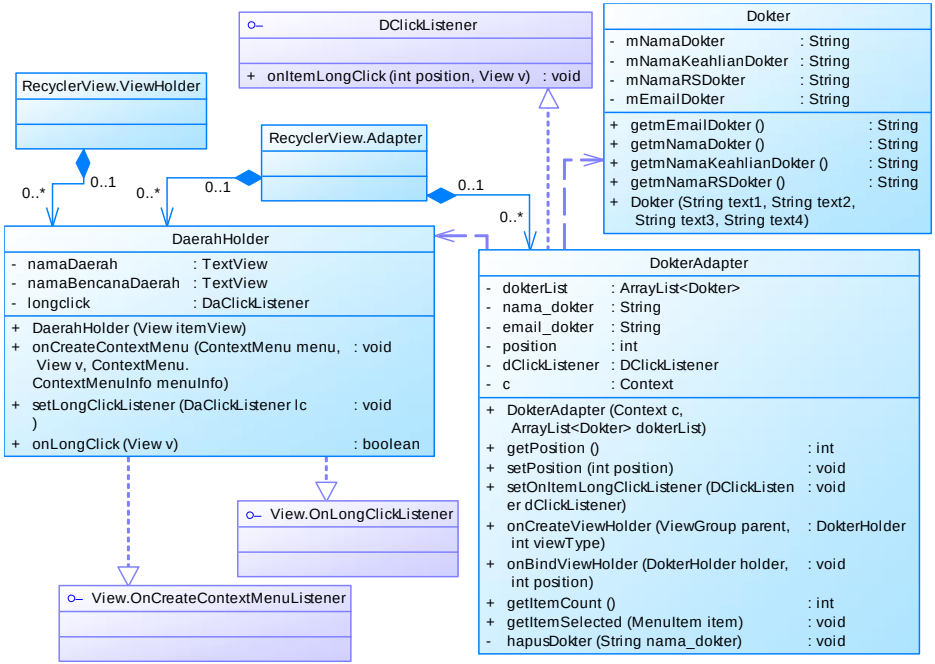
utama yang mempunyai atribut-atribut, yaitu id rumah sakit, nama rumah sakit, dan alamat rumah sakit.



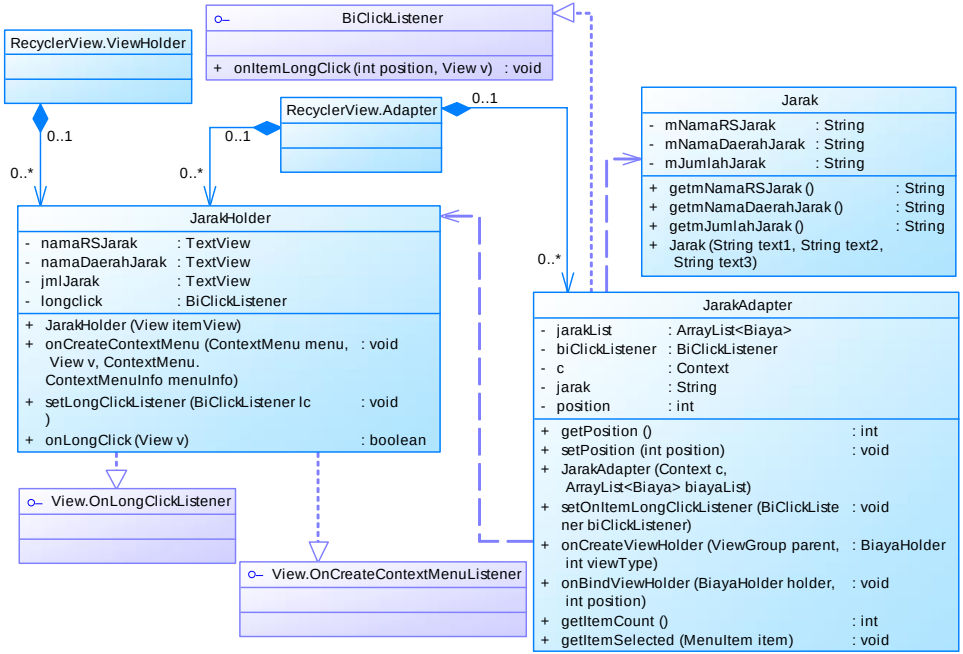
Gambar 3.3 Diagram Kelas Daerah



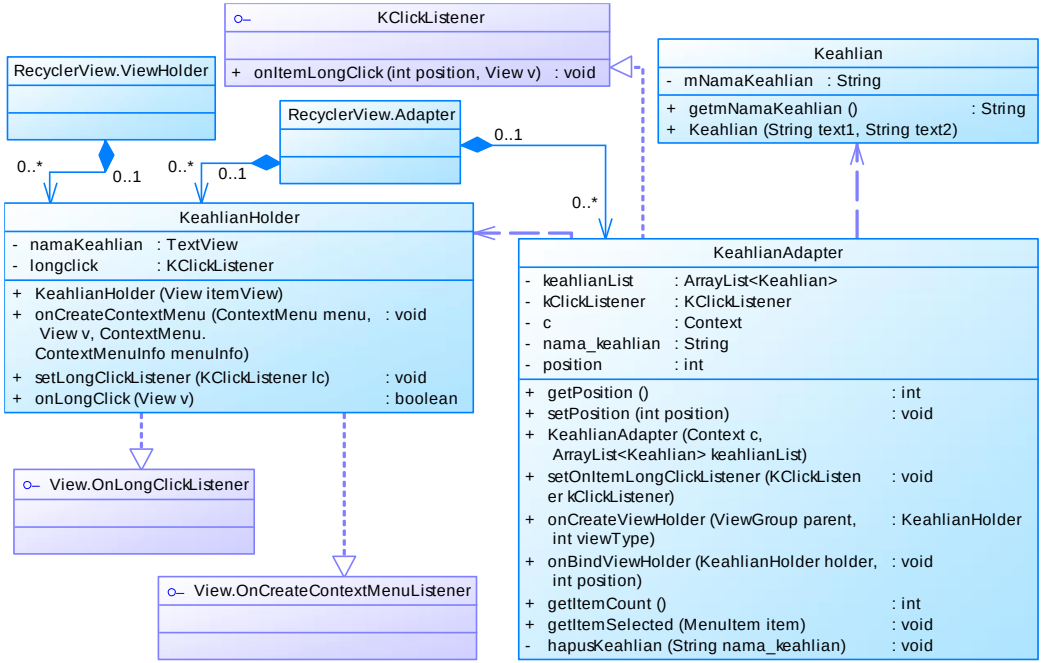
Gambar 3.4 Diagram Kelas Bencana



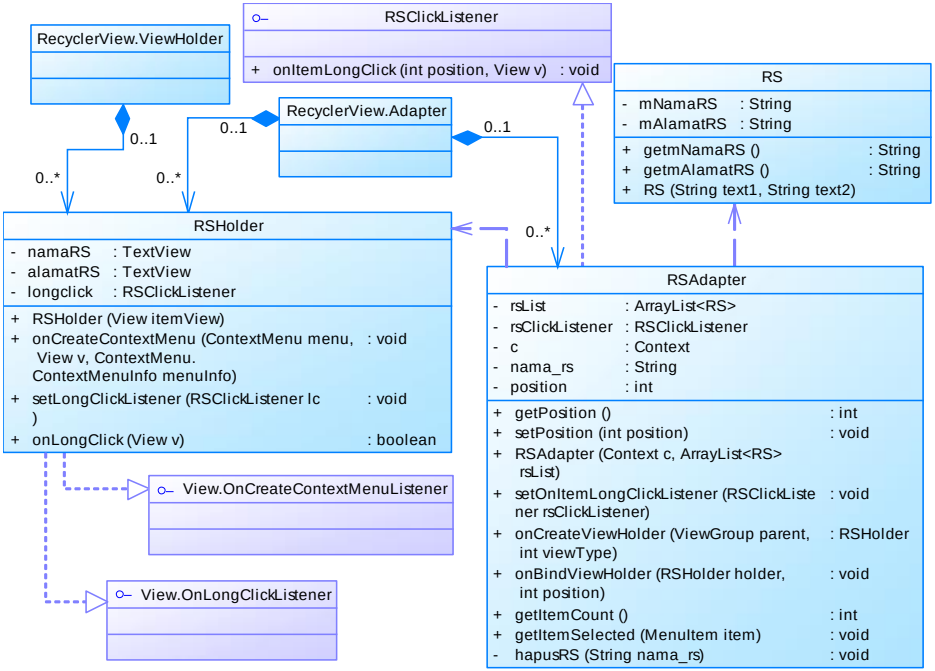
Gambar 3.5 Diagram Kelas Dokter



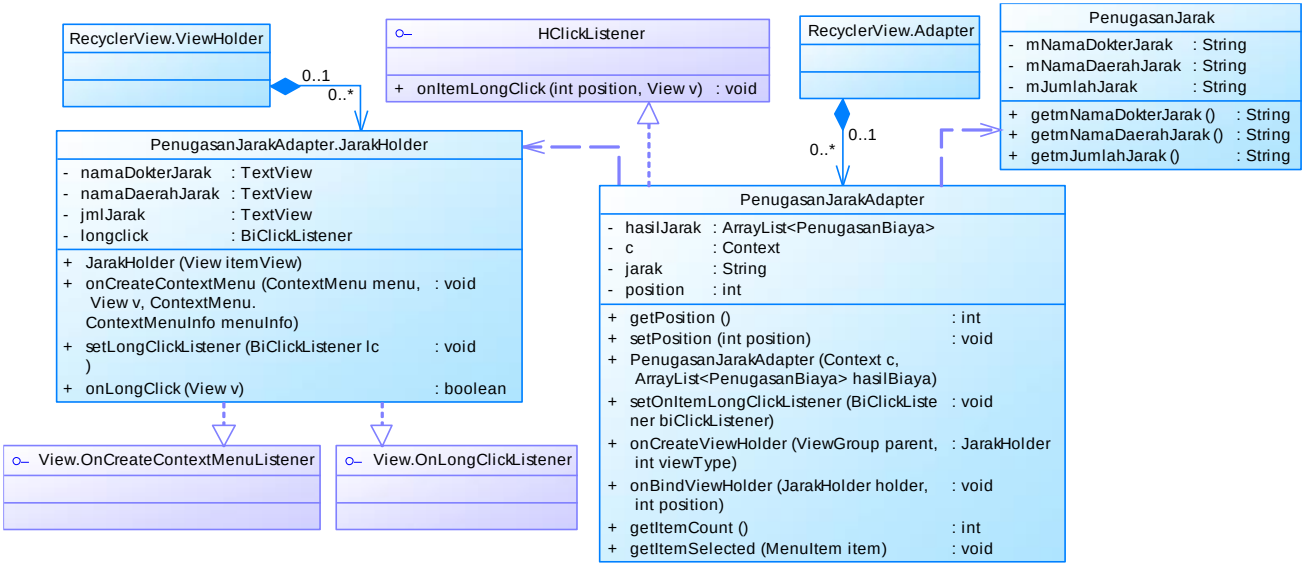
Gambar 3.6 Diagram Kelas Jarak



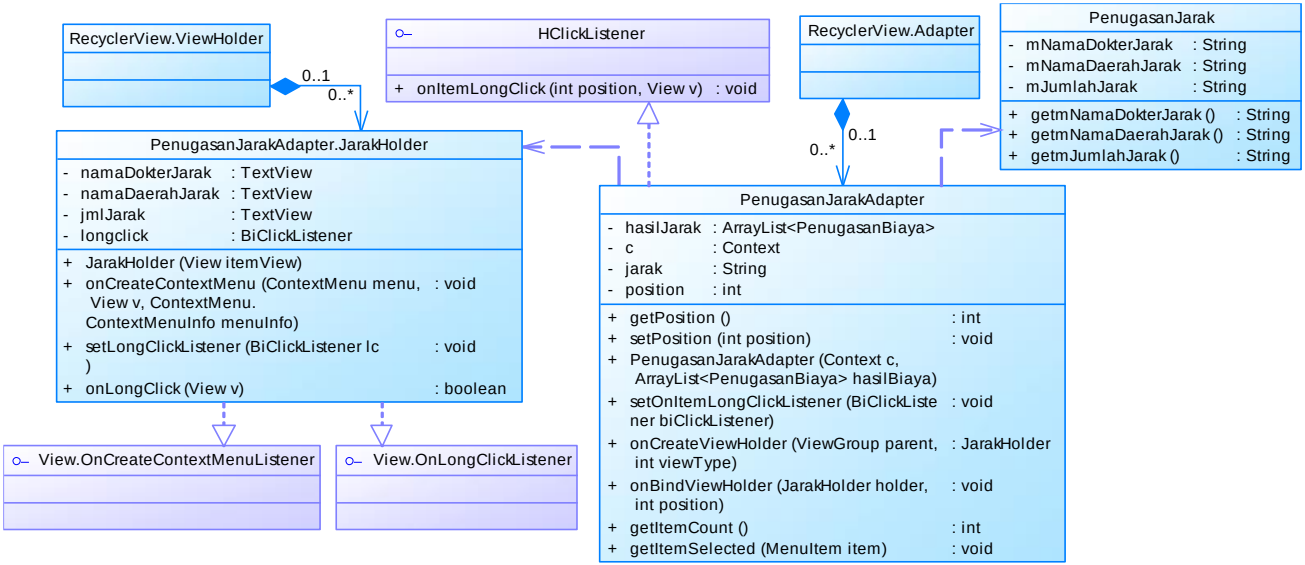
Gambar 3.7 Diagram Kelas Keahlian



Gambar 3.8 Diagram Kelas Rumah Sakit

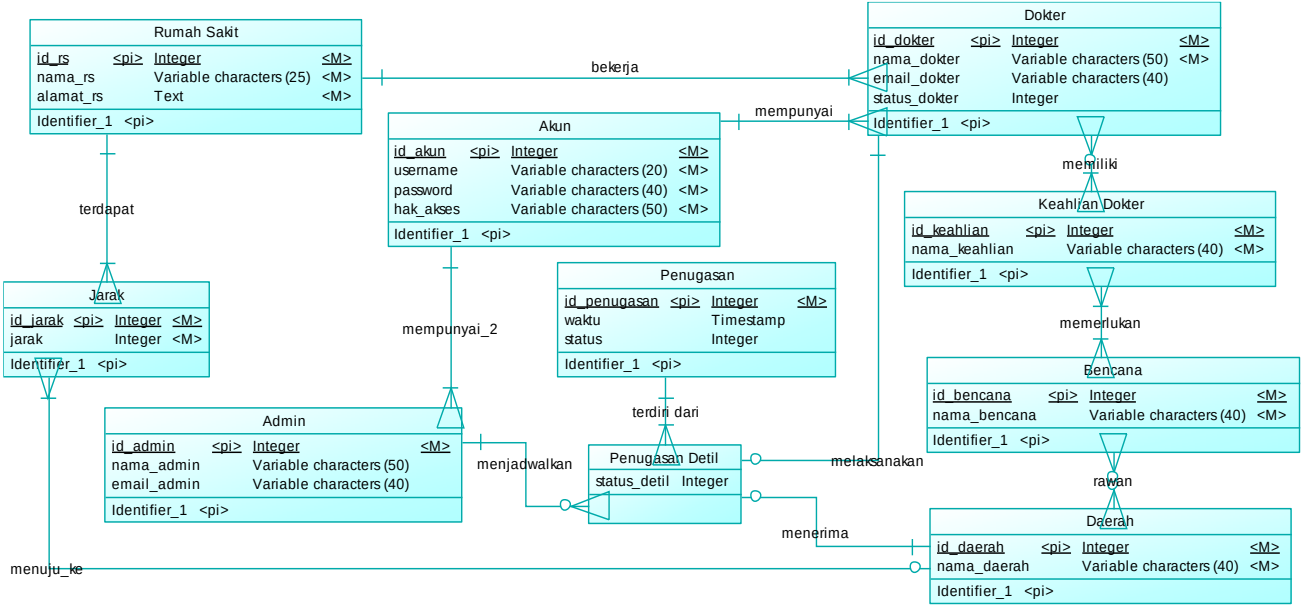


Gambar 3.9 Diagram Kelas Penugasan



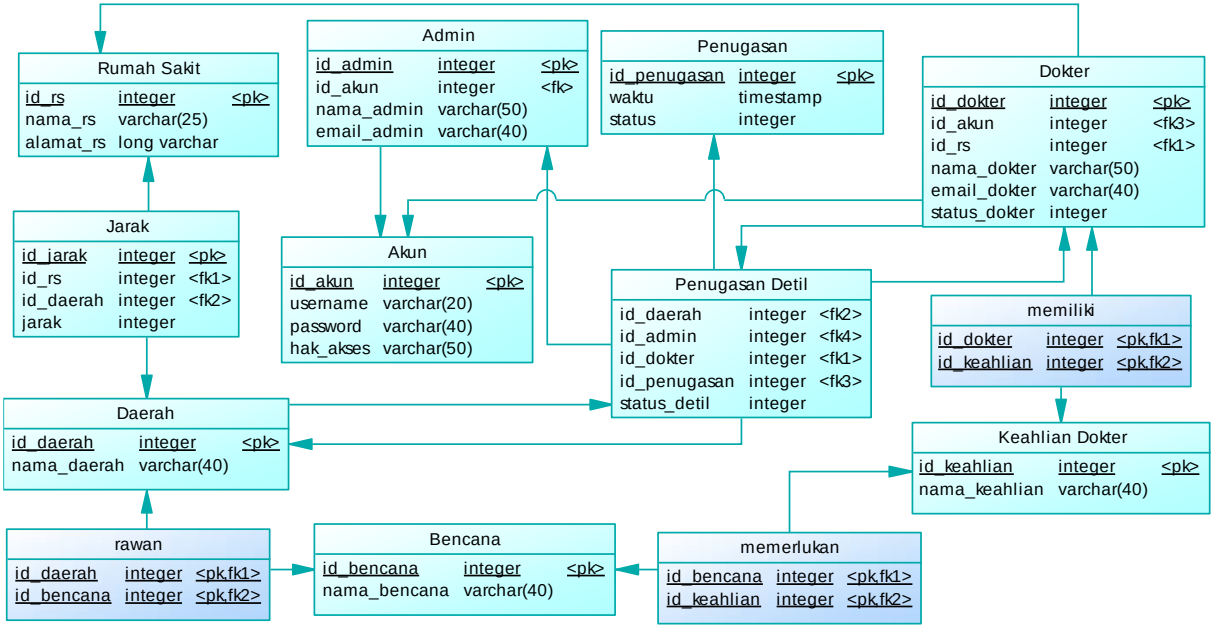
Gambar 3.10 Diagram Kelas Daftar Penugasan

3.2.4.14*Conceptual Data Modeling* (CDM)



Gambar 3.11 Conceptual Data Modeling

3.2.4.15 *Physical Data Modeling* (PDM)



Gambar 3.12 Physical Data Modeling

3.2.5 Perancangan Antarmuka Pengguna

Perancangan antarmuka pengguna merupakan hal yang penting dalam melakukan perancangan perangkat lunak. Antarmuka pengguna yang berhubungan langsung dengan aktor harus memiliki kemudahan-kemudahan dan tampilan yang menarik bagi penggunanya. Aplikasi ini memiliki dua hak akses, yaitu admin dan dokter dengan halaman *login* pengguna yang sama. Pada hak akses admin, aplikasi ini memiliki beberapa antarmuka, yaitu halaman penugasan, penugasan hasil, penugasan ulang, dan riwayat progres penugasan. Pada hak akses dokter, terdapat halaman daftar penugasan, riwayat progres penugasan, dan konfirmasi penugasan selesai.

3.2.5.1 Rancangan Halaman Antarmuka Login Pengguna

Halaman ini digunakan oleh pengguna untuk masuk ke sistem. Pengguna harus mengisi isian *username* dan *password* untuk dapat masuk ke dalam sistem. Rancangan halaman dapat dilihat pada Gambar 0.59. Atribut antarmuka dapat dilihat pada Tabel 3.34.

Tabel 3.34 Atribut Antarmuka Login Pengguna

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*edittextUsername*	*EditText*	Input untuk memasukkan *username* pengguna	*String*
2	*edittextPassword*	*EditText*	Input untuk memasukkan *password* pengguna	*String*
3	*buttonLogin*	*Button*	Tombol aksi untuk menuju ke halaman	*ButtonClick*

			utama admin ataupun dokter	

3.2.5.2 Rancangan Halaman Antarmuka Penugasan

Halaman ini akan ditampilkan pada sistem ketika admin akan menjadwalkan penugasan dokter ke daerah bencana. Rancangan halaman dapat dilihat pada Gambar 0.60. Atribut antarmuka dapat dilihat pada Tabel 3.35.

Tabel 3.35 Atribut Antarmuka Penugasan

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*ButtonPilih Dokter*	*Button*	Tombol aksi untuk mengakses daftar pilihan dokter yang hendak ditugaskan dalam bentuk *checkbox*	*ButtonClick*
2	*buttonPilihDaerah*	*Button*	Tombol aksi untuk mengakses daftar pilihan daerah yang memiliki kondisi darurat (bencana) dalam bentuk *checkbox*	*ButtonClick*
3	*buttonSubmit*	*Button*	Tombol aksi untuk menuju ke halaman penugasan hasil	*ButtonClick*

3.2.5.3 Rancangan Halaman Antarmuka Penugasan Hasil

Halaman ini akan ditampilkan pada hak akses admin setelah admin melakukan penugasan dokter ke daerah bencana. Halaman ini menampilkan penugasan dokter yang optimal ke daerah bencana. Rancangan halaman dapat dilihat pada Gambar 0.61. Atribut antarmuka dapat dilihat pada Tabel 3.36.

Tabel 3.36 Atribut Antarmuka Penugasan Hasil

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*textviewTotalCost*	*TextView*	Informasi total jarak penugasan	*String*
2	*textviewNamaDaerah*	*TextView*	Informasi nama daerah yang memiliki kondisi darurat (bencana)	*String*
3	*textviewNamaDokter*	*TextView*	Informasi nama dokter yang ditugaskan	*String*
4	*textviewJarak*	*TextView*	Informasi jarak penugasan dokter ke daerah bencana	*String*
5	*buttonSimpan*	*Button*	Tombol aksi untuk menyimpan penugasan dokter yang hasilnya telah diperoleh dan ditampilkan tersebut	*ButtonClick*

3.2.5.4 Rancangan Halaman Antarmuka Riwayat Progres Penugasan Admin

Halaman ini akan ditampilkan sistem pada hak akses admin setelah sistem menyimpan penugasan dokter ke daerah bencana yang dilakukan oleh admin. Rancangan halaman dapat dilihat pada Gambar 0.62. Atribut antarmuka dapat dilihat pada Tabel 3.37.

Tabel 3.37 Atribut Antarmuka Riwayat Progres Penugasan Admin

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*textviewWaktu*	*TextView*	Informasi waktu penugasan	*String*
2	*textviewNama Daerah*	*TextView*	Informasi nama daerah mana saja yang akan ditangani pada penugasan tersebut	*String*
3	*textviewStatus*	*TextView*	Informasi status penugasan	*String*

3.2.5.5 Rancangan Halaman Antarmuka Penugasan Ulang

Halaman ini akan ditampilkan pada sistem ketika admin akan menjadwalkan penugasan dokter ke daerah bencana. Rancangan halaman dapat dilihat pada Gambar 0.63. Atribut antarmuka dapat dilihat pada Tabel 3.38.

Tabel 3.38 Atribut Antarmuka Penugasan Ulang

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*textviewDaera*	*TextView*	Informasi nama	*String*

	h		daerah yang jadwal penugasannya diulang	
2	*ButtonPilih Dokter*	*Button*	Tombol aksi untuk mengakses daftar pilihan dokter pengganti yang hendak ditugaskan dalam bentuk *checkbox*	*ButtonClick*
3	*buttonSubmit*	*Button*	Tombol aksi untuk menuju ke halaman penugasan ulang hasil	*ButtonClick*

3.2.5.6 Rancangan Halaman Antarmuka Daftar Penugasan

Halaman ini akan ditampilkan pada hak akses dokter setelah sistem menyimpan jadwal penugasan dokter ke daerah bencana yang diajukan oleh admin. Rancangan halaman dapat dilihat pada Gambar 0.64. Atribut antarmuka dapat dilihat pada Tabel 3.39.

Tabel 3.39 Atribut Antarmuka Daftar Penugasan

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*textviewNama Daerah*	*TextView*	Informasi nama daerah yang memiliki kondisi darurat (bencana)	*String*
2	*textviewJarak*	*TextView*	Informasi jarak	*String*

			penugasan dokter ke daerah bencana	
3	*textviewWaktu*	*TextView*	Informasi waktu penugasan dokter ke daerah bencana	*String*

3.2.5.7 Rancangan Halaman Antarmuka Riwayat Progres Penugasan Dokter

Halaman ini akan ditampilkan sistem pada hak akses dokter setelah sistem menyimpan penugasan dokter ke daerah bencana yang dilakukan oleh admin. Rancangan halaman dapat dilihat pada Gambar 0.65. Atribut antarmuka dapat dilihat pada Tabel 3.40.

Tabel 3.40 Atribut Antarmuka Riwayat Progres Penugasan Dokter

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*textviewWaktu*	*TextView*	Informasi waktu penugasan	*String*
2	*textviewNama Daerah*	*TextView*	Informasi nama daerah mana yang sedang ditangani	*String*
3	*textviewJarak*	*TextView*	Informasi jarak penugasan	*String*
4	*textviewStatus*	*TextView*	Informasi status penugasan	*String*

3.2.5.8 Rancangan Halaman Antarmuka Konfirmasi Penugasan Selesai

Halaman ini akan ditampilkan sistem pada hak akses dokter ketika penugasan sedang berlangsung dan dokter akan memberikan konfirmasi penugasan selesai. Rancangan halaman

dapat dilihat pada Gambar 0.66. Atribut antarmuka dapat dilihat pada Tabel 3.41.

Tabel 3.41 Atribut Antarmuka Konfirmasi Penugasan Selesai

No.	Nama Atribut Antarmuka	Jenis Atribut	Kegunaan	Masukan/ Keluaran
1	*textviewWaktu*	*TextView*	Informasi waktu penugasan	*String*
2	*textviewNama Daerah*	*TextView*	Informasi nama daerah penugasan yang akan dikonfirmasi telah selesai ditangani	*String*
3	*textviewJarak*	*TextView*	Informasi jarak penugasan	*String*
4	*textviewStatus*	*TextView*	Informasi status penugasan	*String*
5	*textviewPenugasanSelesai*	*TextView*	Pertanyaan apakah penugasan tersebut sudah selesai	*String*
6	*buttonSudah*	*Button*	Tombol aksi yang menampilkan *alert dialog* konfirmasi penugasan selesai	*ButtonClick*

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Sebelum penjelasan implementasi akan ditunjukkan terlebih dahulu lingkungan untuk melakukan implementasi.

Pada bagian implementasi ini juga akan dijelaskan mengenai fungsi-fungsi yang digunakan dalam program tugas akhir ini dan disertai dengan kode sumber masing-masing fungsi utama.

4.1 Lingkungan Implementasi Perangkat Lunak

Spesifikasi perangkat keras serta perangkat lunak yang digunakan dalam tahap implementasi perangkat lunak tugas akhir ini seperti dijelaskan pada

Tabel 4.1.

Tabel 4.1 Lingkungan Implementasi Perangkat Lunak

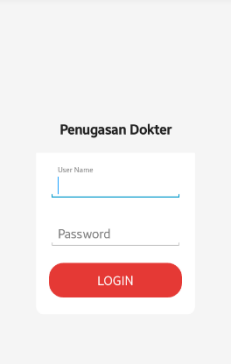
Perangkat Keras	Komputer	HP Pavilion Sleekbook 15
	Prosesor	Intel® Core™ i5-3230M CPU @ 2.60GHz (2.60GHz)
	Memori Primer	8 GB
	Memori Sekunder	500 GB
Perangkat Lunak	Sistem Operasi	*Windows* 8.1 Pro 64-bit

	Perangkat Lunak	Android Studio v2.2.2, Paint, CorelDraw X7, Sybase PowerDesigner 16.5, Microsoft Word 2013

4.2 Implementasi Antarmuka Pengguna

Implementasi antarmuka pengguna berbasis perangkat bergerak ini menggunakan berkas XML yang dibangun pada lingkungan pengembangan Android Studio. Pada subbab ini akan dijelaskan dan ditampilkan tampilan halaman XML sesuai dengan rancangan antarmuka yang terdapat pada bab III.

4.2.1. Implementasi Halaman Antarmuka Login Pengguna

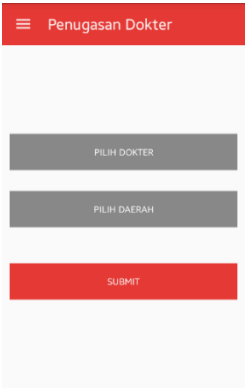


Gambar 4.1 Implementasi Halaman Antarmuka Login Pengguna

Halaman antarmuka login pada Gambar 4.1 merupakan halaman yang digunakan oleh pengguna untuk masuk ke sistem dan mengakses halaman utama. Terdapat tempat isian untuk *username* dan *password*, serta tombol *login*.

4.2.2. Implementasi Halaman Antarmuka Penugasan

Halaman antarmuka penugasan pada Gambar 4.2 merupakan halaman yang akan ditampilkan pada sistem ketika admin akan menjadwalkan penugasan dokter ke daerah bencana.



Gambar 4.2 Implementasi Halaman Antarmuka Penugasan

4.2.3. Implementasi Halaman Antarmuka Penugasan Hasil

Halaman antarmuka penugasan hasil pada Gambar 4.3 merupakan halaman yang akan ditampilkan pada hak akses admin setelah admin melakukan penugasan dokter ke daerah bencana. Halaman ini menampilkan penugasan dokter yang optimal ke daerah bencana.

Gambar 4.3 Implementasi Halaman Antarmuka Penugasan Hasil

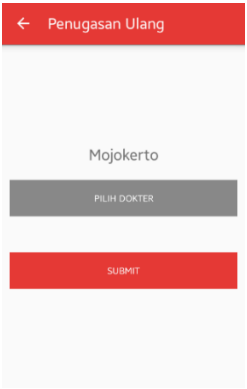
4.2.4. Implementasi Halaman Antarmuka Riwayat Progres Penugasan Admin

Halaman antarmuka riwayat progres penugasan admin pada Gambar 4.4 merupakan halaman yang akan ditampilkan sistem pada hak akses admin setelah sistem menyimpan penugasan dokter ke daerah bencana yang dilakukan oleh admin.

Gambar 4.4 Implementasi Halaman Antarmuka Riwayat Progres Penugasan Admin

4.2.5. Implementasi Halaman Antarmuka Penugasan Ulang

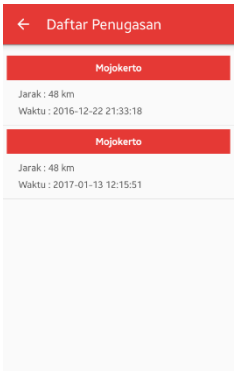
Halaman antarmuka penugasan ulang pada Gambar 4.5 merupakan halaman yang akan ditampilkan pada sistem ketika admin akan menjadwalkan penugasan dokter ke daerah bencana.



Gambar 4.5 Implementasi Halaman Antarmuka Penugasan Ulang

4.2.6. Implementasi Halaman Antarmuka Daftar Penugasan

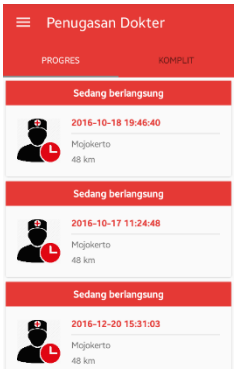
Halaman antarmuka daftar penugasan pada Gambar 4.6 merupakan halaman yang akan ditampilkan pada hak akses dokter setelah sistem menyimpan jadwal penugasan dokter ke daerah bencana yang diajukan oleh admin.



Gambar 4.6 Implementasi Halaman Antarmuka Daftar Penugasan

4.2.7. Implementasi Halaman Antarmuka Riwayat Progres Penugasan Dokter

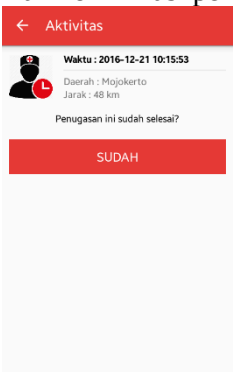
Halaman antarmuka riwayat progres penugasan dokter pada Gambar 4.7 merupakan halaman yang akan ditampilkan sistem pada hak akses dokter setelah sistem menyimpan penugasan dokter ke daerah bencana yang dilakukan oleh admin.



Gambar 4.7 Implementasi Halaman Antarmuka Riwayat Progres Penugasan Dokter

4.2.8. Implementasi Halaman Antarmuka Konfirmasi Penugasan Selesai

Halaman antarmuka konfirmasi penugasan selesai pada Gambar 4.8 merupakan halaman yang akan ditampilkan sistem pada hak akses dokter ketika penugasan sedang berlangsung dan dokter akan memberikan konfirmasi penugasan selesai.



Gambar 4.8 Implementasi Halaman Antarmuka Konfirmasi Penugasan Selesai

4.3 Implementasi Kasus Penggunaan

Implementasi kasus penggunaan aplikasi *mobile* ini menggunakan bahasa pemrograman Java yang dibangun pada lingkungan pengembangan Android Studio. Di sisi pengembangan *web service*, bahasa yang digunakan adalah PHP yang dibangun dengan kerangka kerja CodeIgniter. Pada subbab ini akan menjelaskan dan menampilkan kode sumber pada sisi aplikasi perangkat bergerak pada beberapa kasus penggunaan.

4.3.1 Implementasi Kasus Penggunaan Mengatur Penugasan Dokter

Pada kasus penggunaan ini, digunakan modifikasi Algoritma *Ford-Fulkerson* pada Kode Sumber 4-1, Metode Penugasan Optimal untuk *Minimization Case* pada Kode

Sumber 4-2 dan Metode Penugasan Optimal untuk *Unbalanced Assignment Problem* pada Kode Sumber 4-5 dengan penjelasan masing-masing dapat dilihat pada Tabel 4.2, Tabel 4.3, dan Tabel 4.6. Untuk kode sumber pendukung dan penjelasannya dapat dilihat pada Kode Sumber 4-3, Kode Sumber 4-4, Tabel 4.4 dan Tabel 4.5.

```
1.  private  void getDataJarak(){
2.       StringRequest stringRequest = new
    StringRequest(Request.Method.POST,
    ConfigJson.GET_JARAK_PENUGASAN,
                 new Response.Listener<String>() {
3.                   @Override
4.                   public void onResponse(String
    response) {
5.                        try {
6.                             JSONObject jsonObject= new
    JSONObject(response);
7.                             total_dokter =
    jsonObject.getInt("total_dokter");
8.                             total_daerah =
    jsonObject.getInt("total_daerah");
9.                             nama_dokter = new
    String[total_dokter];
10.                            nama_daerah = new
    String[total_daerah];
11.                            jarak_full = new
    int[total_dokter][total_daerah];
12.                            jarak_binary = new
    int[total_dokter][total_daerah];
13.                            jarak_selected = new
    int[total_dokter][total_daerah];
14.                            JSONArray jsonArray =
    jsonObject.getJSONArray("result");
15.                            for(int
    i=0;i<total_dokter;i++){
16.                                 JSONObject data =
    jsonArray.getJSONObject(i);
17.                                 nama_dokter[i] =
    data.getString("nama_dokter");
18.                                 JSONArray
    daftar keahlian =
    data.getJSONArray("daftar_keahlian");
19.                                 JSONArray jarak_daerah
    = data.getJSONArray("jarak_daerah");
20.                                 for(int
    j=0;j<jarak_daerah.length();j++) {
21.                                      JSONObject
```

```
    data daerah = jarak daerah.getJSONObject(j);
22.                                 JSONArray
    butuh_keahlian =
    data_daerah.getJSONArray("butuh_keahlian");
23.                                  nama_daerah[j] =
    data_daerah.getString("nama_daerah");
24.
25.                                 String b_keahlian
    = "";
26.                                 for(int
    bk=0;bk<butuh keahlian.length();bk++){
27.                                     b keahlian =
    b_keahlian+"|"+butuh_keahlian.getString(bk);
28.                                 }
29. //memasangkan daerah dengan dokter (Modifikasi
    Metode Ford Fulkerson)
30.                                 int count = 0;
31.                                 for(int
    dk=0;dk<daftar_keahlian.length();dk++){
32. if(b_keahlian.contains(daftar_keahlian.getString(d
    k))){
33. count++;}}
34. if(count==butuh_keahlian.length()){
35. jarak_binary[i][j] = 1;
36. jarak_selected[i][j] =
    data_daerah.getInt("jarak");
37. }else{
38. jarak_binary[i][j] = 0;
39. jarak_selected[i][j] = MAX;
40. }
41. jarak_full[i][j] = data_daerah.getInt("jarak");
42.                             }
43.                         }
44.                         oneAssignment();
45.                     } catch (JSONException e) {
46.                         e.printStackTrace();
47.                     }
48.                 }
49.             },
50.             new Response.ErrorListener() {
51.                 @Override
52.                 public void
    onErrorResponse(VolleyError error) {
53. Toast.makeText(PenugasanHasil.this,"Error :
    ("+ResponseStatus+")" +
    error.toString(),Toast.LENGTH_LONG).show();
54.                 }
55.             }){
56.         @Override
57.         protected Map<String,String> getParams(){
```

```
58.             Map<String,String> params = new
    HashMap<String, String>();
59.             params.put("daftar_dokter",
    daftar_dokter);
60.             params.put("daftar_daerah",
    daftar_daerah);
61.             return params;
62.         }
63.     };}
```

Kode Sumber 4-1 Mengambil Data Jarak dari Masukan Dokter dan Daerah yang Sudah Diperoleh

Tabel 4.2 Penjelasan Kode Sumber 4-1

No. Baris	Kegunaan
2	Mengirim request data ke server berupa string
4	Menerima respon berupa string dari server
6	Memperoleh respon data dalam JSON Object
7	Mengambil total dokter dari variabel bertipe JSON Object
8	Mengambil total daerah dari variabel bertipe JSON Object
9	Instansiasi array string nama dokter sebanyak total dokter
10	Instansiasi array string nama daerah sebanyak total daerah
11	Instansiasi matriks integer jarak full dengan ukuran total dokter x total daerah
12	Instansiasi matriks integer jarak binary dengan ukuran total dokter x total daerah
13	Instansiasi matriks integer jarak selected dengan ukuran total dokter x total daerah
14	Instansiasi JSON Array untuk memperoleh data array result dari variabel JSON Object
16-19	Sebanyak total dokter mengambil nama dokter ke dalam array string nama dokter, mengambil array daftar keahlian ke variabel bertipe JSON Array, dan mengambil array jarak daerah ke variabel bertipe JSON Array

21-23	Mengambil objek dari jarak daerah ke dalam variabel data daerah bertipe JSON Object, array keahlian yang dibutuhkan dari data daerah ke dalam variabel bertipe JSON Array, dan mengambil nama daerah ke dalam array string nama daerah
25-33	Menggabungkan semua keahlian yang dibutuhkan pada variabel string b_keahlian dan melakukan pengecekan jika keahlian dari array daftar keahlian terdapat pada keahlian yang dibutuhkan, diberi bobot *count*
34-36	Jika bobot *count* sama dengan banyaknya keahlian yang dibutuhkan dari variabel butuh_keahlian, jarak dari data daerah dimasukkan ke dalam jarak selected
37-40	Jika bobot *count* sama dengan banyaknya keahlian yang dibutuhkan dari variabel butuh_keahlian, jarak selected menyimpan nilai maksimum
41	Mengambil jarak dari data daerah ke dalam jarak full
44	Memanggil fungsi oneAssignment

```
1.  public void oneAssignment(){
2.      pdialog = new
    ProgressDialog(PenugasanHasil.this);
3.      pdialog.setMessage("Menghitung...");
4.      pdialog.setCanceledOnTouchOutside(false);
5.      pdialog.show();
6.      ArrayList<int[]> dokter_col = new
    ArrayList<int[]>();
7.      final_daerah = new int[total_daerah];
8.      final_dokter = new int[total_daerah];
9.      for(int i=0;i<total_daerah;i++) {
10.         final_daerah[i] = i;
11.         int min = 9999;
12.         for(int j=0;j<total_dokter;j++) {
13.             if (jarak_selected[j][i]<min &&
    jarak_selected[j][i]!=MAX){
14.                 min = jarak_selected[j][i];
15.             }
16.         }
17.         int min_count = 0;
18.         for(int j=0;j<total_dokter;j++) {
19.             if (jarak_selected[j][i]==min &&
    jarak_selected[j][i]!=MAX){
20.                 min_count++;
21.             }
```

```
22.          }
23.          int[] dokter s;
24.          if(min_count==0){
25.              min_count=1;
26.              dokter_s = new int[min_count];
27.              dokter s[0]=-1;
28.          }else{
29.              dokter_s = new int[min_count];
30.              int count = 0;
31.
32.              for(int j=0;j<total_dokter;j++) {
33.                  if (jarak_selected[j][i]==min &&
    jarak_selected[j][i]!=MAX){
34.                      dokter_s[count] = j;
35.                      count++;
36.                  }
37.              }
38.          }
39.          dokter_col.add(i,dokter_s);
40.      }
41.      int[] temp_check = new int[total_dokter];
42.      int end = 0;
43.      for(int i=0;i<dokter_col.size();i++) {
44.          int[] temp = dokter col.get(i);
45.          if(temp.length>1){
46.              i=dokter_col.size()+1;
47.          }else{
48.              int unique = 1;
49.              for (int k=0;k<i;k++){
50.                  if(temp_check[k]==temp[0]){
51.                      unique = 0;
52.                  }
53.              }
54.              if(unique==0)
55.                  i=dokter col.size()+1;
56.          }
57.
58.          if(i==dokter_col.size()-1)
59.              end=1;
60.      }
61.      if(end==1){
62.
63.          for(int x=0;x<total_dokter;x++) {
64.              for(int y=0;y<total_daerah;y++) {
65.                  jarak_selected[x][y] = MAX;
66.              }
67.          }
68.          for(int x=0;x<dokter_col.size();x++){
69.              int[] temp_final = dokter_col.get(x);
70.              final_dokter[x] = temp_final[0];
```

```
71.         }
72.
73.         total_cost=0;
74.         for(int x=0;x<total_daerah;x++){
75.             total_cost = total_cost +
    jarak_full[final_dokter[x]][final_daerah[x]];
76.         }
77.         if(pdialog!=null && pdialog.isShowing()){
78.             pdialog.dismiss();
79.             pdialog = null;
80.         }
81.         totalcost.setText("Total cost :
    "+total_cost);
82. viewResult(final_daerah,final_dokter);
83.     }else{
84.         int final_count = 0;
85.         int final_end = 0;
86.         while(final_end==0){
87.             int unique_activity=1;
88.             int i = 0;
89.             int no_unique = 0;
90.             while(unique_activity==1) {
91.                 int[] temp = dokter_col.get(i);
92.                 if (temp.length == 1 && temp[0]!=-
    1) {
93.                     int unique = 1;
94.                     for (int q = 0; q <
    final_dokter.length; q++) {
95.                         if (q != i) {
96.                             int[] temp_2 =
    dokter_col.get(q);
97.                             if (temp_2.length ==
    1) {
98.                                 if (temp_2[0] ==
    temp[0]) {
99.                                     unique = 0;
100.                                }
101.                            } else {
102.                                for (int r = 0;
    r < temp_2.length; r++) {
103.                                    if (temp[0]
    == temp_2[r]) {
104.                                        unique =
    0;
105.                                    }
106.                                }
107.                            }
108.                        }
109.                    }
110.                    if (unique == 1) {
```

```
111.
112.                             final_dokter[i] =
    temp[0];
113.                             final_count++;
114.
                               for (int j = 0; j <
    total_daerah; j++) {
115.
    jarak_selected[j][temp[0]] = MAX;
116.                             }
117.                             for (int k = 0; k <
    total_dokter; k++) {
118.
    jarak_selected[final_dokter[k]][k] = MAX;
119.                             }
120.
121.                             dokter_col.clear();
122.                             for (int x = 0; x <
    total_daerah; x++) {
123.                                 int min = MAX;
124.                                 for (int j = 0; j <
    total_dokter; j++) {
125.                                     if
    (jarak_selected[x][j] < min &&
    jarak_selected[x][j] != MAX) {
126.                                         min =
    jarak_selected[x][j];
127.                                     }
128.                                 }
129.                                 int min_count = 0;
130.                                 int[] dokter_s;
131.                                 if(min!=MAX){
132.                                     for (int j = 0;
    j < total_dokter; j++) {
133.                                         if
    (jarak_selected[x][j] == min &&
    jarak_selected[x][j] != MAX) {
134.
    min_count++;
135.                                         }
136.                                     }
137.                                     dokter_s = new
    int[min_count];
138.                                     int count = 0;
139.                                     for (int j = 0;
140.  j < total_dokter; j++) {
141.                                         if
    (jarak_selected[x][j] == min &&
    jarak_selected[x][j] != MAX) {
142.  dokter_s[count] = j;
```

```
143.                                                    count++;
144.                                                }
145.                                            }
146.                                        }else{
147.                                            dokter_s = new
    int[1];
148.                                            dokter_s[0] = -
    1;
149.                                        }
150.                                        dokter_col.add(x,
    dokter s);
151.                                    }
152.                                    i = -1;
153.                                } else {
154.                                    Log.i("oneAssignment",
    "dokter" + i + " is not unique");
155.                                }
156.                            }
157.                            i++;
158.                            if(i==dokter_col.size()){
159.                                no unique = 1;
160.                                unique activity = 0;
161.                                i=0;
162.                            }
163.
164.                        }
165.                        if(no_unique==1){
166.                          int step5=1;
167.                          int f = 0;
168.                          while(step5==1){
169.                                int[] temp_5 =
    dokter_col.get(f);
170.                                int count_5 = 0;
171.                                if(temp_5.length==1 &&
    temp 5[0]!=-1 && temp 5[0]!=0){
172.                                    for(int
    f2=0;f2<dokter_col.size();f2++){
173.                                        if(f2!=f){
174.                                            int[] temp_f2 =
    dokter col.get(f2);
175.  if(temp_f2.length==1 && temp_f2[0]==temp_5[0]){
176.                                                count_5++;
177.                                            }
178.                                        }
179.                                    }
180.                                }
181.                                if(count_5>0){
182.                                    int[] same_dokter = new
    int[count_5+1];
183.                                    int count_=0;
```

```
184.                            for(int
    f3=0;f3<total_daerah;f3++){
185.                                int[] temp_f3 =
    dokter_col.get(f3);
186.                                if(temp_f3.length==1
    && temp f3[0]==temp 5[0] && temp 5[0]!=-1){
187.  same_dokter[count_] = f3;
188.
189.                                    count_++;
190.                                }
191.                            }
192.                            int[] diff_min_same =
    new int[count_5+1];
193.                            for(int
    f4=0;f4<diff_min_same.length;f4++){
194.                                int[] temp_jarak =
    new int[total_dokter];
195.                                for(int
    f5=0;f5<total_dokter;f5++){
196.                                    temp_jarak[f5] =
    jarak_selected[same_dokter[f4]][f5];
197.                                }
198.  Arrays.sort(temp jarak);
199.                                diff min same[f4] =
    temp_jarak[1] - temp_jarak[0];
200.                            }
201.                        int max_diff = -1;
202.                        int selected_dokter=0;
203.                        for(int
    f6=0;f6<diff_min_same.length;f6++){
204.  if(diff_min_same[f6]>max_diff){
205.                                    selected_dokter
    = same dokter[f6];
206.                                    max diff =
    diff min same[f6];
207.                                }
208.                            }
209.  final_dokter[selected_dokter] = temp_5[0];
210.                            for (int j = 0; j <
    total_daerah; j++) {
211.  jarak_selected[j][final_dokter[selected_dokter]]
    = MAX;
212.                            }
213.                            for (int k = 0; k <
    total_dokter; k++) {
214.  jarak_selected[selected dokter][k] = MAX;
215.                            }
216.
217.                            dokter col.clear();
218.                            for(int
```

```
    x=0;x<total_daerah;x++) {
219.                                int min = MAX;
220.                                for (int j = 0; j <
    total_dokter; j++) {
221.                                    if
    (jarak_selected[x][j] < min &&
    jarak_selected[x][j] != MAX) {
222.                                        min =
    jarak_selected[x][j];
223.                                    }
224.                                }
225.                                int min_count = 0;
226.                                int[] dokter_s;
227.                                if(min!=MAX){
228.                                    for (int j = 0;
229.  j < total_dokter; j++) {
230.                                        if
    (jarak_selected[x][j] == min &&
    jarak_selected[x][j] != MAX) {
231.  min_count++;
232.                                        }
233.
234.                                    }
235.                                    dokter s = new
    int[min_count];
236.                                    int count = 0;
237.                                    for (int j = 0;
    j < total_dokter; j++) {
238.                                        if
    (jarak_selected[x][j] == min &&
    jarak_selected[x][j] != MAX) {
239.
    dokter s[count] = j;
240.                                            count++;
241.                                        }
242.                                    }
243.                                }else{
244.                                    dokter_s = new
    int[1];
245.                                    dokter s[0] = -
    1;
246.                                }
247.                                dokter col.add(x,
    dokter s);
248.                            }
249.                            step5=0;
250.                            unique_activity = 0;
251.                        }else if(count_5==0 &&
    temp 5[0]!=-1 && temp 5[0]!=0){
252.
```

```
                                final_dokter[f] =
    temp 5[0];
253.
                                for (int j = 0; j <
    total_daerah; j++) {
254.
    jarak_selected[j][final_dokter[final_dokter[f]]] =
    MAX;
255.                                }
256.                                for (int k = 0; k <
    total_dokter; k++) {
257.                                    jarak_selected[f][k]
    = MAX;
258.                                }
259.                                dokter_col.clear();
260.                                for(int
    x=0;x<total_daerah;x++) {
261.                                    int min = MAX;
262.                                    for (int j = 0; j <
    total_dokter; j++) {
263.                                        if
    (jarak_selected[x][j] < min &&
    jarak_selected[x][j] != MAX) {
264.                                            min =
    jarak_selected[x][j];
265.                                        }
266.                                    }
267.                                    int min_count = 0;
268.                                    int[] dokter_s;
269.                                    if(min!=MAX){
270.                                        for (int j = 0;
    j < total_dokter; j++) {
271.                                            if
    (jarak_selected[x][j] == min &&
    jarak_selected[x][j] != MAX) {
272.  min_count++;
273.                                            }
274.                                        }
275.                                        dokter_s = new
    int[min_count];
276.                                        int count = 0;
277.                                        for (int j = 0;
278.  j < total_dokter; j++) {
279.                                            if
    (jarak_selected[x][j] == min &&
    jarak_selected[x][j] != MAX) {
280.  dokter_s[count] = j;
281.                                                count++;
282.                                            }
283.                                        }
```

```
284.                                    }else{
285.                                        dokter s = new
    int[1];
286.                                        dokter_s[0] = -
    1;
287.                                    }
288.                                    dokter_col.add(x,
    dokter_s);
289.                                }
290.                                step5=0;
291.                                unique activity = 0;
292.                            }
293.                            f++;
294.                            if(f==dokter_col.size()){
295.                                step5 = 0;
296.                            }
297.                        }
298.                }
299.                int total_max=0;
300.                for(int x=0;x<total_daerah;x++) {
301.                    for(int y=0;y<total_dokter;y++)
    {
302.                        if
    (jarak_selected[x][y]==MAX){
303.                            total_max = total_max+1;
304.                        }
305.                    }
306.                }
307.                if((total_daerah*total_dokter)-
    total_max<=total_daerah){
308.
                      int check=0;
309.                    for(int
    x=0;x<dokter col.size();x++){
310.                        int[] temp_f =
    dokter_col.get(x);
311.                        if(temp_f[0]==0
    ||temp_f==null || temp_f.length==0){
312.                            int count=0;int d=0;
313.                            for(int
    y=0;y<total_dokter;y++){
314.  if(jarak_selected[x][y]==MAX)
315.                                    count++;
316.                                else
317.                                    d = y;
318.                            }
319.                            if(count==total_dokter-
    1)
320.                                check=d;
321.                        }
```

```
322.                     }
323.                     int count=0;
324.                     for(int x=0;x<total_daerah;x++)
    {
325.  if(jarak_selected[x][check]!=MAX){
326.                             count++;
327.                         }
328.                     }
329.  if(count==((total_daerah*total_dokter)-
    total_max)){
330.                         final end=1;
331.                         total_cost = -1;
332.                         totalcost.setText("Tidak
    ditemukan solusi");
333.                         if(pdialog!=null &&
    pdialog.isShowing()){
334.                             pdialog.dismiss();
335.                             pdialog = null;
336.                         }
337.                     }
338.                 }
339.  if(total_max>=(total_daerah*total_dokter)){
340.                     final end=1;
341.                     total_cost=0;
342.                     for(int x=0;x<total_daerah;x++){
343.                         total_cost = total_cost +
    jarak_full[final_daerah[x]][final_dokter[x]];
344.                     }
345.                     totalcost.setText("Total cost :
    "+total_cost);
346.                 }}}}
```

Kode Sumber 4-2 Menghitung Penugasan

Tabel 4.3 Penjelasan Kode Sumber 4-2

No. Baris	Kegunaan
6	Instansiasi kolom dokter pada variabel bertipe ArrayList, dokter_col, yang berisi integer
7	Instansiasi array integer pada variabel final daerah sebanyak total daerah
8	Instansiasi array integer pada variabel final daerah sebanyak total daerah
10-	Menemukan jarak paling minimum yang dimiliki tiap

22	daerah pada kolom daerah yang direpresentasikan pada array integer final daerah.
23-40	Memperoleh dokter yang memiliki jarak paling minimum tiap daerah
41-82	Menemukan jarak yang paling optimal di mana tiap daerah tepat memiliki satu dokter dengan jarak minimum dan menghapus jarak yang sudah optimal dari matriks jarak selected dengan memasukkan nilai maksimum, serta menjumlahkan semua jarak ke dalam variabel total cost
83-164	Solusi optimal di mana tiap daerah tepat memiliki satu dokter dengan jarak minimum belum ditemukan, dilakukan kembali pencarian jarak paling minimum dari tiap daerah yang belum memiliki solusi optimal
165-251	Melakukan pengecekan jika beberapa daerah memiliki jarak minimum pada dokter yang sama untuk dihitung perbedaan jarak dokter tersebut dengan jarak minimum lainnya pada dokter lain terhadap daerah tersebut. Daerah yang dapat menugaskan dokter tersebut adalah daerah dengan perbedaan antar jarak minimumnya yang paling maksimum.
252-258	Menghapus jarak daerah yang sudah memperoleh penugasan dokter yang optimal dengan memasukkan nilai maksimum pada matriks jarak selected
259-266	Melakukan update dokter_col dan menemukan kembali jarak minimum daerah yang belum memperoleh penugasan dokter optimal
267-289	Memperoleh dokter yang memiliki jarak paling minimum tiap daerah yang belum mendapatkan penugasan dokter optimal
299-306	Menghitung banyaknya jarak yang sudah dihapus pada matriks jarak selected dan menyimpannya pada variabel total max
307-338	Mengecek apakah masih terdapat daerah yang belum memperoleh penugasan dokter yang optimal
339-345	Menghitung total jarak penugasan jika semua daerah sudah mendapatkan penugasan dokter

```
1.  public void viewResult(int[]
    data_daerah,int[]data_dokter){
2.      String[] daerah = new
    String[data_daerah.length];
3.      String[] dokter = new
    String[data_dokter.length];
4.      String[] hasil = new
    String[data_daerah.length];
5.      for(int i=0; i<data_daerah.length;i++){
6.          daerah[i] = nama_daerah[data_daerah[i]];
7.          dokter[i] = nama_dokter[data_dokter[i]];
8.          hasil[i] =
    String.valueOf(jarak_full[data_dokter[i]][data_dae
    rah[i]]);
9.          result_daerah = result_daerah + daerah[i]
    + "|";
10.         result_dokter = result_dokter + dokter[i]
    + "|";
11.     }
12.     result_daerah = result_daerah.substring(0,
    result_daerah.length()-1);
13.     result_dokter = result_dokter.substring(0,
    result_dokter.length()-1);
14.     for(int i=0;i<daerah.length;i++){
15.         PenugasanJarak post = new
    PenugasanJarak(dokter[i], daerah[i],hasil[i]);
16.         hasilJarak.add(post);
17.     }    mAdapter.notifyDataSetChanged();}
```

Kode Sumber 4-3 Mengatur Hasil Penghitungan Penugasan

Tabel 4.4 Penjelasan Kode Sumber 4-3

No. Baris	Kegunaan
2-4	Inisiasi array string daerah, dokter, dan hasil
5-8	Memasukkan string nama daerah pada array string daerah, string nama dokter pada array string dokter, dan jarak pada array string hasil dari matriks jarak full, serta
9-10	Menggabungkan tiap string dari array daerah pada variabel string result daerah dan tiap string dari array dokter pada variabel string result dokter

15	Instansiasi PenugasanJarak dari data dokter, daerah, dan hasil yang sudah diperoleh ke variabel post
16	Memasukkan objek PenugasanJarak pada variabel post ke variabel bertipe ArrayList hasilJarak

```
1.  public void onSimpanHasil(View view){
2.          StringRequest stringRequest = new
    StringRequest(Request.Method.POST,
    ConfigJson.ADD_DATA_PENUGASAN,
                    new Response.Listener<String>() {
3.                      @Override
4.                      public void onResponse(String
    response) {
5.                          try {
6.                              JSONObject jsonObject=
    new JSONObject(response.toString());
7.                              ResponseStatus =
    jsonObject.getString("status");
8.                          } catch (JSONException e)
9.  {
10.                             e.printStackTrace();
11.                         }
12. if(ResponseStatus.equals("berhasil")){
13. Toast.makeText(getApplicationContext(),"Data
    Berhasil Disimpan", Toast.LENGTH_SHORT).show();
14.                         }
15.                         else{
16. Toast.makeText(getApplicationContext(),"Simpan
    Hasil Penugasan Gagal",
    Toast.LENGTH_SHORT).show();
17.                         }
18.                     }
19.                 },
20.                 new Response.ErrorListener() {
21.                     @Override
22.                     public void
    onErrorResponse(VolleyError error) {
23.
24. Toast.makeText(PenugasanHasil.this,"Error :
    ("+ResponseStatus+")" +
    error.toString(),Toast.LENGTH_LONG).show();
25.                     }
26.                 }) {
27.             @Override
28.             protected Map<String,String>
    getParams(){
29.                 Map<String,String> params = new
    HashMap<String, String>();
```

```
30.                 params.put("id_akun", id_akun);
31.                 params.put("daftar_dokter",
    result_dokter);
32.                 params.put("daftar_daerah",
    result_daerah);
33.                 return params;            }};}
```

Kode Sumber 4-4 Menyimpan Hasil Penugasan

Tabel 4.5 Penjelasan Kode Sumber 4-4

No. Baris	Kegunaan
2	Mengirim request data ke server berupa string
4	Menerima respon berupa string dari server
6	Instansiasi JSON Object dari respon ke dalam variabel
7	Instansiasi string ResponseStatus dari variabel bertipe JSON Object
12	Mengecek apakah data hasil penghitungan penugasan berhasil disimpan
30	Mengirim parameter id akun
31	Mengirim parameter daftar dokter
32	Mengirim parameter daftar daerah

```
1.  private void unbalanced(){
2.      int matrix_unbalance[][] = jarak_selected;
3.      ArrayList<int[]> col_dokter = new
    ArrayList<int[]>();
4.      int diff_dokter[] = new int[total_daerah];
5.      int end=0,iter=0;
6.          while(end==0){
7.           for(int y=0;y<total_daerah;y++){
8.              int sorted_dokter[] = new
    int[total_daerah];
9.              for(int x=0;x<total_daerah;x++){
10.                 if(matrix_unbalance[x][y]==-1)
11.                     sorted_dokter[x] = MAX*MAX;
12.                 else
13.                     sorted_dokter[x] =
    matrix unbalance[x][y];
14.             }
15.             sorted_dokter =
    BubbleSortAsceMethod(sorted dokter);
16.             while(get_diff==0){
```

```
17.              if(sorted dokter[a]==-
    1||sorted dokter[b]==-1){ a++;b++;
18.              }else{
19.                diff_dokter[y] = sorted_dokter[a]-
    sorted dokter[b];
20.                min cost = sorted dokter[b];
21.                get_diff=1;
22.              }
23.             }
24.             int count=0;
25.             int temp_col_dokter[] = new
    int[total_daerah];
26.             for(int x=0;x<total_daerah;x++){
27.              temp_col_dokter[x]=-1}
28.             for(int x=0;x<total_daerah;x++){
29. if(matrix_unbalance[x][y]==min_cost){
                        temp col dokter[count] = x;
                        count++;
30.                  }
31.             }
32.             col dokter.add(y,temp col dokter);
33.         }
34.         int max_diff[] = new int[total_daerah];
35.         max diff = diff dokter2;
36.         int deleting_row = 0;
37.         max_diff =
    BubbleSortDescMethod(diff dokter);
38.         for(int y=0;y<total_daerah;y++){
39.          if(max_diff[0]==diff_dokter[y]){
40.             deleting_row = y;
41.          }
42.         }
43.         int[] minimum_col =
    col dokter.get(deleting row);
44.         final daerah[iter]=deleting row;
45.         final_dokter[iter]=minimum_col[0];
46.         iter++;
47.         for(int x=0;x<total_daerah;x++){
48.          for(int y=0;y<total_daerah;y++){
49.            if(x==minimum_col[0]||y==deleting_row){
50.                          matrix_unbalance[x][y]=-1;
51.            }
52.           }
53.         }
54.         int count=0;
55.     for(int a=0;a<total_daerah;a++){
56.      for(int b=0;b<total_daerah;b++){
           if(matrix_unbalance[a][b]==-1){
                count++;}}}
57.     if(count==total_daerah*total_daerah||
```

```
        iter>total_daerah){
58.         end=1;
59.         printUnbalance(final_daerah,final_dokter);
60.     }else{
61.         end =0;}
62.     }}}
```

Kode Sumber 4-5 Menghitung Penugasan Unbalance

Tabel 4.6 Penjelasan Kode Sumber 4-5

No. Baris	Kegunaan
2-4	Instansiasi matriks pada variabel matrix_unbalance, array integer pada variabel diff_dokter, dan arraylist pada variabel col_dokter
7-23	Mencari dan menghitung selisih dua jarak minimum tiap daerah. Kemudian, memasukkan jarak minimum pada variabel min_cost
24-32	Memasukkan indeks dokter dengan jarak minimum pada col_dokter
34-37	Mencari selisih jarak minimum terbesar dan memasukkannya ke dalam variabel array max_diff
38-53	Menghapus baris daerah dan indeks dokter yang memiliki selisih jarak minimum terbesar pada col_dokter dengan mengubah nilai jaraknya pada matrix_unbalance menjadi -1
54-61	Mengecek apakah semua daerah sudah memperoleh solusi penugasan di mana data jarak pada matrix_unbalance nya = -1. Jika belum, proses penghitungan dilakukan kembali.

4.3.2 Implementasi Kasus Penggunaan Mengatur Penugasan Ulang Dokter

Kode sumber dan penjelasannya dapat dilihat pada Kode Sumber 4-1, Kode Sumber 4-2, Kode Sumber 4-6, Kode Sumber 4-7, Tabel 4.2, Tabel 4.3, Tabel 4.7, dan Tabel 4.8.

```
1.  private void getDokterPengganti(){
```

```
2.      StringRequest stringRequest = new
    StringRequest(Request.Method.POST,
    ConfigJson.GET_DOKTER_PENGGANTI,
                new Response.Listener<String>() {
3.                  @Override
4.                  public void onResponse(String
    response) {
5.                      try {
6.                          JSONObject jsonObject =
    new JSONObject(response);
7.                          JSONArray Result =
    jsonObject.getJSONArray("data");
8.                          int size =
    Result.length();
9.                          for (int i = 0; i < size;
    i++) {
10.                             JSONObject obj result
    = (JSONObject) Result.get(i);
11.                             String nama_dokter =
    obj_result.getString("nama_dokter");
12.                             results_dokter.add(i,
    nama dokter);
13.                         dokter_arr = new
    String[results_dokter.size()];
14.                         dokter_arr =
    results_dokter.toArray(dokter_arr);
15.                     } catch (JSONException e) {
16.                         e.printStackTrace();
17.                     }
18.                 }
19.             },
20.             new Response.ErrorListener() {
21.                 @Override
22.                 public void
    onErrorResponse(VolleyError error) {
23. Toast.makeText(PenugasanUlang.this,"Error :
    ("+ResponseStatus+")" +
    error.toString(),Toast.LENGTH_LONG).show();
24.                 }
25.             }){
26.         @Override
27.         protected Map<String,String> getParams(){
28.             Map<String,String> params = new
    HashMap<String, String>();
29.             params.put("nama_dokter",
    daftar_dokter);
30.             params.put("id_penugasan",
    id_penugasan);
31.             return params;     }};}
```

Kode Sumber 4-6 Mengambill Data Dokter Pengganti

Tabel 4.7 Penjelasan Kode Sumber 4-6

No. Baris	Kegunaan
2	Melakukan request data dokter pengganti ke server berupa string
4	Menerima respon berupa string dari server
6	Instansiasi JSON Object dari respon ke dalam variabel
7	Mengambil JSON Array dari objek respon ke dalam variabel bertipe JSON Array
10	Mengambil JSON Array data dokter pengganti ke dalam variabel bertipe JSON Object
11	Mengambil nama dokter dari variabel bertipe JSON Object
12	Memasukkan nama dokter ke dalam variabel bertipe ArrayList
29	Mengirim parameter nama dokter
30	Mengirim parameter id penugasan

```
1.  public void onSimpanHasil(View view){
2.      StringRequest stringRequest = new
    StringRequest(Request.Method.POST,
    ConfigJson.ADD_DATA_PENUGASAN_ULANG,
                new Response.Listener<String>() {
3.                  @Override
4.                  public void onResponse(String
    response) {
5.                      try {
6.                          JSONObject jsonObject= new
    JSONObject(response.toString());
7.                          ResponseStatus =
    jsonObject.getString("status");
8.                      } catch (JSONException e) {
9.                          e.printStackTrace();
10.                     }
11. if(ResponseStatus.equals("berhasil")){
12. Toast.makeText(getApplicationContext(),"Data
```

```
    Berhasil Disimpan", Toast.LENGTH_SHORT).show();
13.                     }
14.                     else{
15. Toast.makeText(getApplicationContext(),"Simpan
    Hasil Penugasan Gagal",
    Toast.LENGTH_SHORT).show();
16.                     }
17.                 }
18.             },
19.             new Response.ErrorListener() {
20.                 @Override
21.                 public void
    onErrorResponse(VolleyError error) {
22. Toast.makeText(PenugasanUlangHasil.this,"Error :
    ("+ResponseStatus+")" +
    error.toString(),Toast.LENGTH_LONG).show();
23.                 }
24.             }) {
25.         @Override
26.         protected Map<String,String> getParams(){
27.             Map<String,String> params = new
    HashMap<String, String>();
28.             params.put("id_akun", id_akun);
29.             params.put("id_penugasan",
    id_penugasan);
30.             params.put("daftar_dokter",
    result_dokter);
31.             params.put("daftar_daerah",
    result_daerah);
32.             params.put("dokter_old",
    nama_dokter_old);
33.             return params; }
34.     };}
```

Kode Sumber 4-7 Menyimpan Hasil Penugasan Ulang

Tabel 4.8 Penjelasan Kode Sumber 4-7

No. Baris	Kegunaan
2	Mengirim request data ke server berupa string
4	Menerima respon berupa string dari server
6	Instansiasi JSON Object dari respon ke dalam variabel
7	Instansiasi string ResponseStatus dari variabel bertipe JSON Object

11	Mengecek apakah data hasil penghitungan penugasan berhasil disimpan
28	Mengirim parameter id akun
29	Mengirim parameter id penugasan
30	Mengirim parameter daftar dokter
31	Mengirim parameter daftar daerah
32	Mengirim parameter dokter old

4.3.3 Implementasi Kasus Penggunaan Melihat Daftar Penugasan

Kode sumber dan penjelasannya dapat dilihat pada Kode Sumber 4-8 dan Tabel 4.9.

```
1.  public ArrayList<DaftarPenugasanItem>
    getDataPenugasan(){
2.      StringRequest stringRequest = new
    StringRequest(Request.Method.POST,
    ConfigJson.DAFTAR_PENUGASAN,
                new Response.Listener<String>() {
3.                  @Override
4.                  public void onResponse(String
    response) {
5.                      try {
6.                          JSONObject jsonObject= new
    JSONObject(response.toString());
7.                          JSONArray jsonArray =
    jsonObject.getJSONArray("data");
8.                          for(int
    i=0;i<jsonArray.length();i++){
9.                              JSONObject data =
    (JSONObject) jsonArray.get(i);
10.                             String nama_daerah =
    data.getString("nama_daerah");
11.                             String jarak =
    data.getString("jarak");
12.                             String waktu =
    data.getString("waktu");
13.                             String id_penugasan =
    data.getString("id_penugasan");
14.                             DaftarPenugasanItem
    post = new DaftarPenugasanItem(nama_daerah, jarak,
    waktu,id_penugasan);
15. penugasanList.add(post);
16.                             DaftarPenugasanItem
```

```
        obj = new DaftarPenugasanItem(nama daerah, jarak,
        waktu,id penugasan);
17.                                 results.add(i, obj);}
18. mAdapter.notifyDataSetChanged();
19. if(jsonArray.length()==0){
20. penugasan_kosong.setVisibility(View.VISIBLE);
21.                             }
22.                         } catch (JSONException e) {
23.                             e.printStackTrace();
24.                         }}},
25.                 new Response.ErrorListener() {
26.                     @Override
27.                     public void
        onErrorResponse(VolleyError error) {
28. Toast.makeText(DaftarPenugasan.this,"Error :
        ("+ResponseStatus+")" +
        error.toString(),Toast.LENGTH_LONG).show();
29.                     }
30.                 }){
31.             @Override
32.             protected Map<String,String> getParams(){
33.                 Map<String,String> params = new
        HashMap<String, String>();
34.                 params.put("id_akun",
        shared.getString("user_id","0"));
35.                 return params; }};
36. return results;}
```

Kode Sumber 4-8 Mengambil Data Penugasan

Tabel 4.9 Penjelasan Kode Sumber 4-8

No. Baris	Kegunaan
2	Mengirim request data ke server berupa string
4	Menerima respon berupa string dari server
6	Instansiasi JSON Object dari respon ke dalam variabel
7	Mengambil JSON Array dari objek respon ke dalam variabel bertipe JSON Array
9	Mengambil JSON Array data dokter pengganti ke dalam variabel bertipe JSON Object
10	Mengambil nama daerah dari variabel bertipe JSON Object

11	Mengambil jarak dari variabel bertipe JSON Object
12	Mengambil waktu dari variabel bertipe JSON Object
13	Mengambil id penugasan dari variabel bertipe JSON Object
14, 16	Instansiasi objek DaftarPenugasanItem dari nama daerah, jarak, waktu, dan id penugasan yang sudah diperoleh
15	Memasukkan hasil instansiasi objek DaftarPenugasanItem ke dalam variabel bertipe List
17	Memasukkan hasil instansiasi objek DaftarPenugasanItem ke dalam variabel bertipe ArrayList
34	Mengirim parameter id_akun

4.3.4 Implementasi Kasus Penggunaan Menyetujui Penugasan

Kode sumber dan penjelasannya dapat dilihat pada Kode Sumber 4-9 dan Tabel 4.10.

```
1.  public void simpanKonfirmasi(final String
    id_penugasan, final String status){
2.      StringRequest stringRequest = new
    StringRequest(Request.Method.POST,
    ConfigJson.KONFIRMASI_PENUGASAN,
                new Response.Listener<String>() {
3.                  @Override
4.                  public void onResponse(String
    response) {
5.                      try {
6.                          JSONObject jsonObject= new
    JSONObject(response.toString());
7.  if(jsonObject.getString("status").equals("berhasil
    ")){
8.  if(status.equals("1")){
9.  Toast.makeText(DaftarPenugasan.this, "Penugasan
    berhasil dipilih", Toast.LENGTH_LONG).show();
10.                                 startActivity(new
    Intent(DaftarPenugasan.this,Aktivitas.class));
11.                             }
12.                             else
    if(status.equals("2"))
13. Toast.makeText(DaftarPenugasan.this, "Penugasan
    tidak dipilih", Toast.LENGTH_LONG).show();
```

```
14.                             else
15. Toast.makeText(DaftarPenugasan.this, "Penugasan
    gagal dikonfirmasi", Toast.LENGTH_LONG).show();
16.
                                }
17.                         } catch (JSONException e) {
18.                             e.printStackTrace();
19.                         }
20.                     }
21.                 },
22.                 new Response.ErrorListener() {
23.                     @Override
24.                     public void
    onErrorResponse(VolleyError error) {
25. Toast.makeText(DaftarPenugasan.this,"Error :
    ("+ResponseStatus+")" +
    error.toString(),Toast.LENGTH_LONG).show();
26.                     }             }){
27.         @Override
28.         protected Map<String,String> getParams(){
29.             Map<String,String> params = new
    HashMap<String, String>();
30.             params.put("id_penugasan",
    id penugasan);
31.             params.put("id_akun",
    shared.getString("user_id","0"));
32.             params.put("status", status);
33.             return params; }};}
```

Kode Sumber 4-9 Menyimpan Konfirmasi Penugasan

Tabel 4.10 Penjelasan Kode Sumber 4-9

No. Baris	Kegunaan
2	Mengirim request data ke server berupa string
4	Menerima respon berupa string dari server
6	Instansiasi JSON Object dari respon ke dalam variabel
7	Mengecek apakah data konfirmasi penugasan berhasil disimpan
8-11	Menampilkan pesan bahwa penugasan disetujui
12-13	Menampilkan pesan bahwa penugasan ditolak
30	Mengirim parameter id penugasan

31	Mengirim parameter id akun
32	Mengirim parameter status

4.3.5 Implementasi Kasus Penggunaan Menolak Penugasan

Kode sumber dan penjelasannya dapat dilihat pada Kode Sumber 4-9 dan Tabel 4.10.

4.3.6 Implementasi Kasus Penggunaan Memberikan Konfirmasi Penugasan Selesai

Kode sumber dan penjelasannya dapat dilihat pada Kode Sumber 4-10 dan Tabel 4.11.

```
1.  public void selesaiPenugasanExec(){
2.      StringRequest stringRequest = new
    StringRequest(Request.Method.POST,
    ConfigJson.SELESAI_PENUGASAN,
                new Response.Listener<String>() {
3.                  @Override
4.                  public void onResponse(String
    response) {
5.                      try {
6.                          JSONObject jsonObject= new
    JSONObject(response.toString());
7.  if(jsonObject.getString("status").equals("berhasil
    ")){
8.  Toast.makeText(Aktivitas.this, "Laporan telah
    terkirim", Toast.LENGTH_LONG).show();
9.                                  startActivity(new
    Intent(Aktivitas.this,MainActivity.class));
10.                         }
11.                     } catch (JSONException e) {
12.                         e.printStackTrace();
13.                     }
14.                 }
15.             },
16.             new Response.ErrorListener() {
17.                 @Override
18.                 public void
    onErrorResponse(VolleyError error) {
19. Toast.makeText(Aktivitas.this,"Error :
    ("+ResponseStatus+")" +
    error.toString(),Toast.LENGTH_LONG).show();
```

```
20.                     }
21.                 }){
22.             @Override
23.             protected Map<String,String> getParams(){
24.                 Map<String,String> params = new
    HashMap<String, String>();
25.                 params.put("id_akun", id_akun);
26.                 params.put("id_penugasan",
    id_penugasan);
27.                 return params;}};}
```

Kode Sumber 4-10 Menyimpan Konfirmasi Penugasan Selesai

Tabel 4.11 Penjelasan Kode Sumber 4-10

No. Baris	Kegunaan
2	Mengirim request data ke server berupa string
4	Menerima respon berupa string dari server
6	Instansiasi JSON Object dari respon ke dalam variabel
7	Mengecek apakah data konfirmasi penugasan selesai berhasil disimpan
8	Menampilkan pesan bahwa konfirmasi penugasan berhasil disimpan
25	Mengirim parameter id akun
26	Mengirim parameter id penugasan

[Halaman ini sengaja dikosongkan]

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas tentang pengujian dan evaluasi pada perangkat lunak yang dibangun untuk tugas akhir ini. Pengujian dilakukan pada kasus penggunaan dari sistem perangkat lunak.

5.1 Lingkungan Pengujian

Pada proses pengujian perangkat lunak, dibutuhkan suatu lingkungan pengujian yang sesuai dengan standar kebutuhan. Lingkungan pengujian dalam tugas akhir ini dilakukan pada setiap kasus penggunaan. Spesifikasi masing-masing lingkungan pengujian dijabarkan pada Tabel 5.1.

Tabel 5.1 Lingkungan Pengujian Fungsionalitas Perangkat Lunak

Spesifikasi	**Deskripsi**
Jenis Perangkat	*Smartphone*
Merek Perangkat	Samsung Grand Duos GT-I9082
Sistem Operasi	Android Jellybean v4.2.2
Memori Internal	8 GB
RAM	1 GB

5.2 Pengujian Fungsionalitas

Pengujian fungsionalitas ini adalah pengujian fungsi-fungsi yang berjalan pada aplikasi berdasarkan kasus penggunaan. Data-data daerah, rumah sakit, dan jarak penugasan yang digunakan dalam pengujian ini bersumber dari Google Maps [13], sehingga data jarak penugasan dari

rumah sakit tempat dokter bekerja ke daerah bencana dapat mendekati jarak yang sebenarnya. Data jarak ini diperoleh dengan menggunakan fitur "Directions" pada Google Maps dengan satuan kilometer (km). Fitur ini dijalankan dengan menentukan alamat rumah sakit sebagai asal, kemudian menentukan suatu daerah di wilayah Jawa Timur sebagai tujuan. Data jarak sebagai data uji ini dipilih berdasarkan hasil navigasi Google Maps dengan rute tercepat dan rute yang dapat dilalui oleh kendaraan roda empat. Ilustrasi cara perolehan data jarak penugasan untuk pengujian ini dapat dilihat pada Gambar 5.43. Data jarak antara beberapa rumah sakit ke beberapa daerah di Jawa Timur yang digunakan pada pengujian ini terlampir pada Tabel 0.1.

Pengujian fungsionalitas ini dijelaskan pada Tabel 5.2, Tabel 5.3, Tabel 5.4, Tabel 5.5, Tabel 5.6, Tabel 5.7, Tabel 5.8, Tabel 5.9, Tabel 5.10, Tabel 5.11, Tabel 5.12, Tabel 5.13, Tabel 5.14, Tabel 5.15, Tabel 5.16, Tabel 5.17, Tabel 5.18, Tabel 5.19, Tabel 5.20, Tabel 5.21, Tabel 5.22, Tabel 5.23, Tabel 5.24, Tabel 5.25, Tabel 5.26, Tabel 5.27, Tabel 5.28, Tabel 5.29, dan Tabel 5.30.

5.2.1. Pengujian Fungsionalitas Melihat Data Daerah

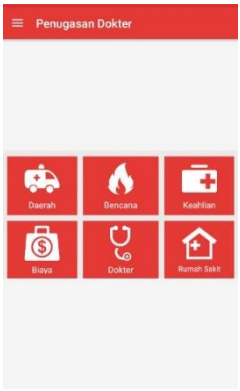
Tabel 5.2 Skenario 1 Pengujian Fungsionalitas Melihat Data Daerah

No. Pengujian	**SCF-001**
Skenario Pengujian	Aplikasi menampilkan data daerah ketika admin masuk ke halaman Kelola Data Daerah
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data yang terdapat beberapa menu kelola data, yaitu "Daerah", "Bencana", "Keahlian", "Rumah Sakit", "Dokter", "Jarak"
Aksi yang dilakukan	Memilih tombol "Daerah"
Hasil yang diharapkan	Aplikasi dapat menampilkan data daerah beserta rawan daerah dari masing-masing daerah

Hasil yang diperoleh	Aplikasi menampilkan data daerah beserta rawan daerah dari masing-masing daerah
Hasil Pengujian	Berhasil

Langkah-langkah SCF-001:

1. Penguji masuk ke halaman halaman Kelola Data seperti pada Gambar 5.1 dan mengisikan semua isian yang terdapat pada halaman tersebut.



Gambar 5.1 Halaman Kelola Data

2. Penguji menekan tombol yang bertuliskan “Daerah”. Aplikasi menampilkan *progress dialog* sebagai tanda bahwa sistem sedang memproses pengambilan data daerah.
3. Aplikasi menampilkan data daerah beserta rawan bencana masing-masing daerah seperti pada Gambar 5.2 yang menunjukkan bahwa penguji telah sukses melihat data daerah

Gambar 5.2 Halaman Kelola Data Daerah

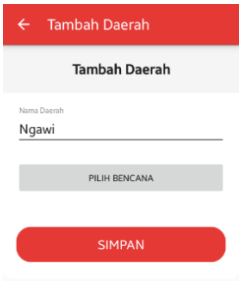
5.2.2. Pengujian Fungsionalitas Menambah Data Daerah

Tabel 5.3 Skenario 1 Pengujian Fungsionalitas Menambah Data Daerah

No. Pengujian	**SCF-002**
Skenario Pengujian	Menambah data daerah untuk disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Tambah Data Daerah yang berisikan isian nama daerah yang masih kosong, tombol yang bertuliskan “Pilih Bencana”, serta tombol yang bertuliskan “Simpan”
Aksi yang dilakukan	Mengisi isian nama daerah dan memilih rawan bencana pada halaman Tambah Data Daerah
Hasil yang diharapkan	Aplikasi dapat menampilkan data daerah yang telah ditambah pada halaman Kelola Data Daerah
Hasil yang diperoleh	Aplikasi menampilkan data daerah yang telah ditambah pada halaman Kelola Data Daerah
Hasil Pengujian	Berhasil

Langkah-langkah SCF-002:

1. Penguji masuk ke halaman Tambah Data Daerah seperti pada Gambar 5.3 dan mengisi isian nama daerah.



Gambar 5.3 Halaman Tambah Data Daerah ketika Mengisi Nama Daerah

2. Penguji memilih rawan bencana dengan memilih tombol "Pilih Bencana" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar rawan bencana yang ada pada sistem.
3. Penguji memilih rawan bencana dengan memilih pilihan "OK" setelah memberikan ceklis pada daftar rawan bencana. Aplikasi menampilkan daftar rawan bencana yang sudah dipilih.
4. Penguji menyimpan data daerah yang ditambah dengan memilih tombol "Simpan". Aplikasi menampilkan semua data daerah termasuk data daerah yang baru ditambah pada halaman Kelola Data Daerah seperti pada Gambar 5.4.

Gambar 5.4 Halaman Kelola Data Daerah setelah Menyimpan Data Daerah yang Ditambah

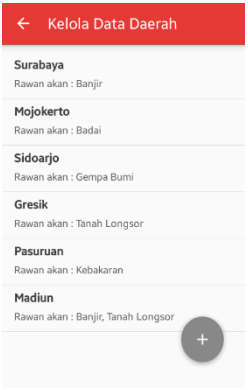
5.2.3. Pengujian Fungsionalitas Mengedit Data Daerah

Tabel 5.4 Skenario 1 Pengujian Fungsionalitas Mengedit Data Daerah

No. Pengujian	**SCF-003**
Skenario Pengujian	Mengedit data daerah dan disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Daerah yang berisi data daerah yang terdiri dari nama daerah dan rawan bencana
Aksi yang dilakukan	Menekan lama bagian data daerah yang hendak diedit
Hasil yang diharapkan	Aplikasi dapat menampilkan data daerah yang telah diedit pada halaman Kelola Data Daerah
Hasil yang diperoleh	Aplikasi menampilkan data daerah yang telah diedit pada halaman Kelola Data Daerah
Hasil Pengujian	Berhasil

Langkah-langkah SCF-003:

1. Penguji masuk ke halaman Kelola Data Daerah dan menekan lama pada bagian data daerah yang hendak diedit sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Edit Data Daerah" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan halaman Edit Data Daerah.
3. Penguji masuk ke halaman Edit Data Daerah dan mengedit nama daerah yang terdapat pada isian nama daerah.
4. Penguji memilih rawan bencana dengan memilih tombol "Pilih Bencana" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang berisi semua daftar rawan bencana. Daftar rawan bencana yang sudah diceklis menunjukkan bahwa bencana tersebut merupakan rawan bencana yang sudah disimpan pada daerah tersebut sebelumnya.
5. Penguji mengedit pilihan rawan bencana dengan memilih pilihan "OK" setelah mengedit ceklis pada daftar rawan bencana. Aplikasi menampilkan daftar rawan bencana yang sudah dipilih.
6. Penguji menyimpan data daerah yang diedit dengan memilih tombol "Simpan". Aplikasi menampilkan semua data daerah termasuk data daerah yang baru diedit pada halaman Kelola Data Daerah seperti pada Gambar 5.5.



Gambar 5.5 Halaman Kelola Data Daerah setelah Menyimpan Data Daerah yang Diedit

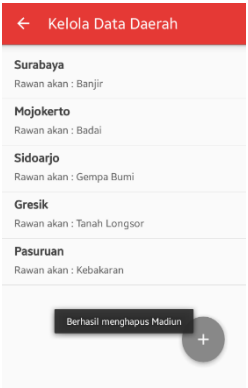
5.2.4. Pengujian Fungsionalitas Menghapus Data Daerah

Tabel 5.5 Skenario 1 Pengujian Fungsionalitas Menghapus Data Daerah

No. Pengujian	**SCF-004**
Skenario Pengujian	Menghapus data daerah dari sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Daerah yang berisi data daerah yang terdiri dari nama daerah dan rawan bencana
Aksi yang dilakukan	Menekan lama bagian data daerah yang hendak dihapus
Hasil yang diharapkan	Aplikasi dapat menampilkan *alert text* bahwa data daerah yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Daerah
Hasil yang diperoleh	Aplikasi menampilkan *alert text* bahwa data daerah yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Daerah
Hasil Pengujian	Berhasil

Langkah-langkah SCF-004:

1. Penguji masuk ke halaman Kelola Data Daerah dan menekan lama pada bagian data daerah yang hendak dihapus sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Hapus Data Daerah" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan *alert dialog* yang menunjukkan data daerah yang dipilih berhasil dihapus dan halaman Kelola Data Daerah seperti pada Gambar 5.6 Halaman Kelola Data Daerah setelah Data Daerah Dihapus



Gambar 5.6 Halaman Kelola Data Daerah setelah Data Daerah Dihapus

5.2.5. Pengujian Fungsionalitas Melihat Data Bencana

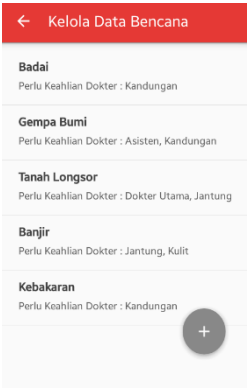
Tabel 5.6 Skenario 1 Pengujian Fungsionalitas Melihat Data Bencana

No. Pengujian	**SCF-005**
Skenario Pengujian	Aplikasi menampilkan data bencana ketika admin masuk ke halaman Kelola Data Bencana

Kondisi Awal	Aplikasi menampilkan halaman Kelola Data yang terdapat beberapa menu kelola data, yaitu “Daerah”, “Bencana”, “Keahlian”, “Rumah Sakit”, “Dokter”, “Jarak”
Aksi yang dilakukan	Memilih tombol “Bencana”
Hasil yang diharapkan	Aplikasi dapat menampilkan data bencana beserta keahlian yang dibutuhkan dari masing-masing bencana
Hasil yang diperoleh	Aplikasi menampilkan data bencana beserta keahlian yang dibutuhkan dari masing-masing bencana
Hasil Pengujian	Berhasil

Langkah-langkah SCF-005:

1. Penguji masuk ke halaman halaman Kelola Data seperti pada Gambar 5.1 dan mengisikan semua isian yang terdapat pada halaman tersebut.
2. Penguji menekan tombol yang bertuliskan “Bencana”. Aplikasi menampilkan *progress dialog* sebagai tanda bahwa sistem sedang memproses pengambilan data bencana.
3. Aplikasi menampilkan data bencana beserta keahlian yang dibutuhkan masing-masing bencana seperti pada Gambar 5.7 yang menunjukkan bahwa penguji telah sukses melihat data bencana



Gambar 5.7 Halaman Kelola Data Bencana

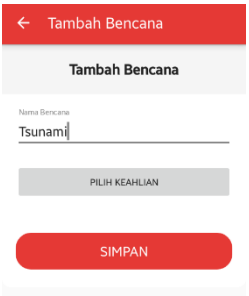
5.2.6. Pengujian Fungsionalitas Menambah Data Bencana

Tabel 5.7 Skenario 1 Pengujian Fungsionalitas Menambah Data Bencana

No. Pengujian	**SCF-006**
Skenario Pengujian	Menambah data bencana untuk disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Tambah Data Bencana yang berisikan isian nama bencana yang masih kosong, tombol yang bertuliskan "Pilih Keahlian", serta tombol yang bertuliskan "Simpan"
Aksi yang dilakukan	Mengisi isian nama bencana dan memilih keahlian yang dibutuhkan pada halaman Tambah Data Bencana
Hasil yang diharapkan	Aplikasi dapat menampilkan data bencana yang telah ditambah pada halaman Kelola Data Bencana
Hasil yang diperoleh	Aplikasi menampilkan data bencana yang telah ditambah pada halaman Kelola Data Bencana
Hasil Pengujian	Berhasil

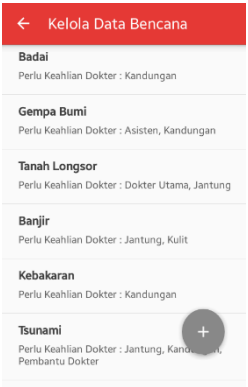
Langkah-langkah SCF-006:

1. Penguji masuk ke halaman Tambah Data Bencana seperti pada Gambar 5.8 dan mengisi isian nama bencana.



Gambar 5.8 Halaman Tambah Data Bencana ketika Mengisi Nama Bencana

2. Penguji memilih keahlian yang dibutuhkan dengan memilih tombol "Pilih Keahlian" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar keahlian yang ada pada sistem.
3. Penguji memilih keahlian yang dibutuhkan dengan memilih pilihan "OK" setelah memberikan ceklis pada daftar keahlian. Aplikasi menampilkan daftar keahlian yang sudah dipilih.
4. Penguji menyimpan data bencana yang ditambah dengan memilih tombol "Simpan". Aplikasi menampilkan semua data bencana termasuk data bencana yang baru ditambah pada halaman Kelola Data Bencana seperti pada Gambar 5.9.



Gambar 5.9 Halaman Kelola Data Bencana setelah Menyimpan Data Bencana yang Ditambah

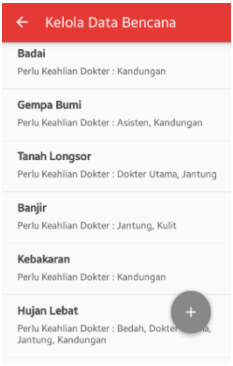
5.2.7. Pengujian Fungsionalitas Mengedit Data Bencana

Tabel 5.8 Skenario 1 Pengujian Fungsionalitas Mengedit Data Bencana

No. Pengujian	**SCF-007**
Skenario Pengujian	Mengedit data bencana dan disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Bencana yang berisi data bencana yang terdiri dari nama bencana dan keahlian yang dibutuhkan
Aksi yang dilakukan	Menekan lama bagian data bencana yang hendak diedit
Hasil yang diharapkan	Aplikasi dapat menampilkan data bencana yang telah diedit pada halaman Kelola Data Bencana
Hasil yang diperoleh	Aplikasi menampilkan data bencana yang telah diedit pada halaman Kelola Data Bencana
Hasil Pengujian	Berhasil

Langkah-langkah SCF-007:

1. Penguji masuk ke halaman Kelola Data Bencana dan menekan lama pada bagian data bencana yang hendak diedit sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Edit Data Bencana" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan halaman Edit Data Bencana.
3. Penguji masuk ke halaman Edit Data Bencana dan mengedit nama bencana yang terdapat pada isian nama bencana.
4. Penguji memilih keahlian yang dibutuhkan dengan memilih tombol "Pilih Keahlian" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang berisi semua daftar keahlian. Daftar keahlian yang sudah diceklis menunjukkan bahwa keahlian tersebut merupakan keahlian yang sudah disimpan pada bencana tersebut sebelumnya.
5. Penguji mengedit pilihan keahlian yang diperlukan dengan memilih pilihan "OK" setelah mengedit ceklis pada daftar keahlian. Aplikasi menampilkan daftar keahlian yang sudah dipilih.
6. Penguji menyimpan data bencana yang diedit dengan memilih tombol "Simpan". Aplikasi menampilkan semua data bencana termasuk data bencana yang baru diedit pada halaman Kelola Data Bencana seperti pada Gambar 5.10.



Gambar 5.10 Halaman Kelola Data Bencana setelah Menyimpan Data Bencana yang Diedit

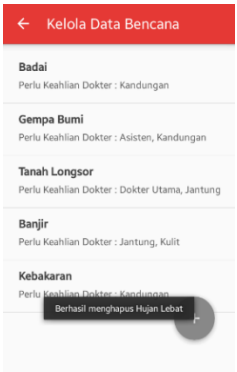
5.2.8. Pengujian Fungsionalitas Menghapus Data Bencana

Tabel 5.9 Skenario 1 Pengujian Fungsionalitas Menghapus Data Bencana

No. Pengujian	**SCF-008**
Skenario Pengujian	Menghapus data bencana dari sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Bencana yang berisi data bencana yang terdiri dari nama bencana dan keahlian yang dibutuhkan
Aksi yang dilakukan	Menekan lama bagian data bencana yang hendak dihapus
Hasil yang diharapkan	Aplikasi dapat menampilkan *alert text* bahwa data bencana yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Bencana
Hasil yang diperoleh	Aplikasi menampilkan *alert text* bahwa data bencana yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Bencana
Hasil Pengujian	Berhasil

Langkah-langkah SCF-008:

1. Penguji masuk ke halaman Kelola Data Bencana dan menekan lama pada bagian data bencana yang hendak dihapus sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Hapus Data Bencana" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan *alert dialog* yang menunjukkan data bencana yang dipilih berhasil dihapus dan halaman Kelola Data Bencana seperti pada Gambar 5.11.



Gambar 5.11 Halaman Kelola Data Bencana setelah Data Bencana Dihapus

5.2.9. Pengujian Fungsionalitas Melihat Data Dokter

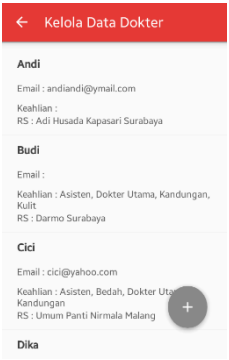
Tabel 5.10 Skenario 1 Pengujian Fungsionalitas Melihat Data Dokter

No. Pengujian	**SCF-009**
Skenario Pengujian	Aplikasi menampilkan data dokter ketika admin masuk ke halaman Kelola Data Dokter
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data yang terdapat beberapa menu kelola data, yaitu "Daerah",

	"Bencana", "Keahlian", "Rumah Sakit", "Dokter", "Jarak"
Aksi yang dilakukan	Memilih tombol "Dokter"
Hasil yang diharapkan	Aplikasi dapat menampilkan data dokter beserta email, keahlian, dan rumah sakit dari masing-masing dokter
Hasil yang diperoleh	Aplikasi menampilkan data dokter beserta email, keahlian, dan rumah sakit dari masing-masing dokter
Hasil Pengujian	Berhasil

Langkah-langkah SCF-009:

1. Penguji masuk ke halaman halaman Kelola Data seperti pada Gambar 5.1 dan mengisikan semua isian yang terdapat pada halaman tersebut.
2. Penguji menekan tombol yang bertuliskan "Dokter". Aplikasi menampilkan *progress dialog* sebagai tanda bahwa sistem sedang memproses pengambilan data dokter.
3. Aplikasi menampilkan data dokter beserta email, keahlian, dan rumah sakit dari masing-masing dokter seperti pada Gambar 5.12 yang menunjukkan bahwa penguji telah sukses melihat data dokter



Gambar 5.12 Halaman Kelola Data Dokter

5.2.10. Pengujian Fungsionalitas Menambah Data Dokter

Tabel 5.11 Skenario 1 Pengujian Fungsionalitas Menambah Data Dokter

No. Pengujian	**SCF-010**
Skenario Pengujian	Menambah data dokter untuk disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Tambah Data Dokter yang berisikan isian nama dokter yang masih kosong, isian email dokter yang masih kosong, tombol yang bertuliskan "Pilih Keahlian", tombol yang bertuliskan "Pilih RS", dan tombol yang bertuliskan "Simpan"
Aksi yang dilakukan	Mengisi isian nama dokter dan email dokter, serta memilih keahlian dokter dan rumah sakit tempat dokter bekerja pada halaman Tambah Data Dokter
Hasil yang diharapkan	Aplikasi dapat menampilkan data dokter yang telah ditambah pada halaman Kelola Data Dokter
Hasil yang diperoleh	Aplikasi menampilkan data dokter yang telah ditambah pada halaman Kelola Data Dokter
Hasil Pengujian	Berhasil

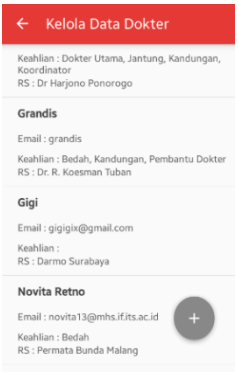
Langkah-langkah SCF-010:

1. Penguji masuk ke halaman Tambah Data Dokter seperti pada Gambar 5.13 dan mengisi isian nama dokter dan email dokter.

Gambar 5.13 Halaman Tambah Data Dokter ketika Mengisi Nama Dokter dan Email Dokter

2. Penguji memilih keahlian dokter dengan memilih tombol "Pilih Keahlian" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar keahlian yang ada pada sistem.
3. Penguji memilih keahlian dokter dengan memilih pilihan "OK" setelah memberikan ceklis pada daftar keahlian. Aplikasi menampilkan daftar keahlian yang sudah dipilih.
4. Penguji memilih rumah sakit tempat dokter bekerja dengan memilih tombol "Pilih RS" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar rumah sakit yang ada pada sistem.
5. Penguji memilih rumah sakit dengan memilih pilihan "OK" setelah memilih satu rumah sakit pada daftar rumah sakit. Aplikasi menampilkan daftar rumah sakit yang sudah dipilih.

6. Penguji menyimpan data dokter yang ditambah dengan memilih tombol "Simpan". Aplikasi menampilkan semua data dokter termasuk data dokter yang baru ditambah pada halaman Kelola Data Dokter seperti pada Gambar 5.14.



Gambar 5.14 Halaman Kelola Data Dokter setelah Menyimpan Data Dokter yang Ditambah

5.2.11. Pengujian Fungsionalitas Mengedit Data Dokter

Tabel 5.12 Skenario 1 Pengujian Fungsionalitas Mengedit Data Dokter

No. Pengujian	**SCF-011**
Skenario Pengujian	Mengedit data dokter dan disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Dokter yang berisi data dokter yang terdiri dari nama, email, dan keahlian dokter, serta rumah sakit tempat dokter bekerja
Aksi yang dilakukan	Menekan lama bagian data dokter yang hendak diedit
Hasil yang diharapkan	Aplikasi dapat menampilkan data dokter yang telah diedit pada halaman Kelola Data Dokter
Hasil yang	Aplikasi menampilkan data dokter yang telah diedit

diperoleh	pada halaman Kelola Data Dokter
Hasil Pengujian	Berhasil

Langkah-langkah SCF-011:

1. Penguji masuk ke halaman Kelola Data Dokter dan menekan lama pada bagian data dokter yang hendak diedit sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Edit Data Dokter" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan halaman Edit Data Dokter.
3. Penguji masuk ke halaman Edit Data Dokter dan mengedit nama dan email dokter yang terdapat pada isian nama dan email dokter.
4. Penguji memilih keahlian dokter dengan memilih tombol "Pilih Keahlian" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang berisi semua daftar keahlian. Daftar keahlian yang sudah diceklis menunjukkan bahwa keahlian tersebut merupakan keahlian yang sudah disimpan pada dokter tersebut sebelumnya.
5. Penguji mengedit pilihan keahlian dokter dengan memilih pilihan "OK" setelah mengedit ceklis pada daftar keahlian. Aplikasi menampilkan daftar keahlian yang sudah.
6. Penguji memilih satu rumah sakit tempat dokter bekerja dengan memilih tombol "Pilih RS" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang berisi semua daftar rumah sakit. Rumah sakit yang sudah ditandai menunjukkan bahwa rumah sakit tersebut merupakan rumah sakit yang sudah disimpan pada dokter tersebut sebelumnya.
7. Penguji mengedit pilihan rumah sakit dengan memilih pilihan "OK" setelah menandai satu rumah sakit pada

daftar rumah sakit. Aplikasi menampilkan rumah sakit yang sudah dipilih.

8. Penguji menyimpan data dokter yang diedit dengan memilih tombol "Simpan". Aplikasi menampilkan semua data dokter termasuk data dokter yang baru diedit pada halaman Kelola Data Dokter seperti pada Gambar 5.15.

Gambar 5.15 Halaman Kelola Data Dokter setelah Menyimpan Data Dokter yang Diedit

5.2.12. Pengujian Fungsionalitas Menghapus Data Dokter

Tabel 5.13 Skenario 1 Pengujian Fungsionalitas Menghapus Data Dokter

No. Pengujian	**SCF-012**
Skenario Pengujian	Menghapus data dokter dari sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Dokter yang berisi data dokter yang terdiri dari nama dokter, email dokter, dan keahlian dokter, serta rumah sakit tempat dokter bekerja
Aksi yang dilakukan	Menekan lama bagian data dokter yang hendak dihapus

Hasil yang diharapkan	Aplikasi dapat menampilkan *alert text* bahwa data dokter yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Dokter
Hasil yang diperoleh	Aplikasi menampilkan *alert text* bahwa data dokter yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Dokter
Hasil Pengujian	Berhasil

Langkah-langkah SCF-012:

1. Penguji masuk ke halaman Kelola Data Dokter dan menekan lama pada bagian data dokter yang hendak dihapus sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Hapus Data Dokter" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan *alert dialog* yang menunjukkan data dokter yang dipilih berhasil dihapus dan halaman Kelola Data Dokter seperti pada Gambar 5.16.

Gambar 5.16 Halaman Kelola Data Dokter setelah Data Dokter Dihapus

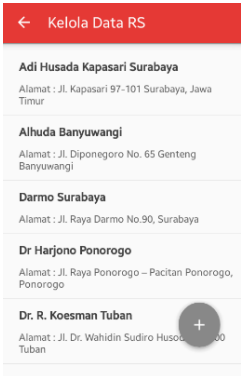
5.2.13. Pengujian Fungsionalitas Melihat Data Rumah Sakit

Tabel 5.14 Skenario 1 Pengujian Fungsionalitas Melihat Data Rumah Sakit

No. Pengujian	**SCF-013**
Skenario Pengujian	Aplikasi menampilkan data rumah sakit ketika admin masuk ke halaman Kelola Data Rumah Sakit
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data yang terdapat beberapa menu kelola data, yaitu "Daerah", "Bencana", "Keahlian", "Rumah Sakit", "Dokter", "Jarak"
Aksi yang dilakukan	Memilih tombol "Rumah Sakit"
Hasil yang diharapkan	Aplikasi dapat menampilkan data rumah sakit beserta alamat dari masing-masing rumah sakit
Hasil yang diperoleh	Aplikasi menampilkan data rumah sakit beserta alamat dari masing-masing rumah sakit
Hasil Pengujian	Berhasil

Langkah-langkah SCF-013:

1. Penguji masuk ke halaman halaman Kelola Data seperti pada Gambar 5.1 dan mengisikan semua isian yang terdapat pada halaman tersebut.
2. Penguji menekan tombol yang bertuliskan "Rumah Sakit". Aplikasi menampilkan *progress dialog* sebagai tanda bahwa sistem sedang memproses pengambilan data rumah sakit.
3. Aplikasi menampilkan data rumah sakit beserta alamat dari masing-masing rumah sakit seperti pada Gambar 5.17 yang menunjukkan bahwa penguji telah sukses melihat data rumah sakit.



Gambar 5.17 Halaman Kelola Data Rumah Sakit

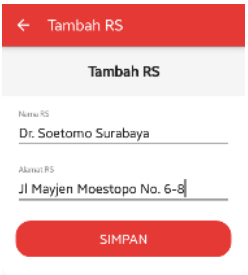
5.2.14. Pengujian Fungsionalitas Menambah Data Rumah Sakit

Tabel 5.15 Skenario 1 Pengujian Fungsionalitas Menambah Data Rumah Sakit

No. Pengujian	**SCF-014**
Skenario Pengujian	Menambah data rumah sakit untuk disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Tambah Data RS yang berisikan isian nama rumah sakit yang masih kosong, isian alamat rumah sakit yang masih kosong, dan tombol yang bertuliskan "Simpan"
Aksi yang dilakukan	Mengisi isian nama rumah sakit dan alamat rumah sakit pada halaman Tambah Data RS
Hasil yang diharapkan	Aplikasi dapat menampilkan data rumah sakit yang telah ditambah pada halaman Kelola Data RS
Hasil yang diperoleh	Aplikasi menampilkan data rumah sakit yang telah ditambah pada halaman Kelola Data RS
Hasil Pengujian	Berhasil

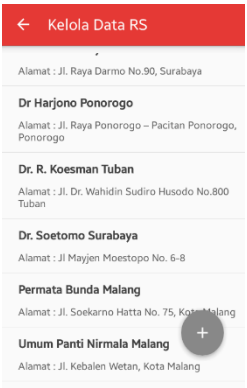
Langkah-langkah SCF-014:

1. Penguji masuk ke halaman Tambah Data RS seperti pada Gambar 5.18 dan mengisi isian nama rumah sakit dan alamat rumah sakit.



Gambar 5.18 Halaman Tambah Data RS ketika Mengisi Nama Rumah Sakit dan Alamat Rumah Sakit

2. Penguji menyimpan data rumah sakit yang ditambah dengan memilih tombol "Simpan". Aplikasi menampilkan semua data rumah sakit termasuk data rumah sakit yang baru ditambah pada halaman Kelola Data RS seperti pada Gambar 5.19.



Gambar 5.19 Halaman Kelola Data RS setelah Menyimpan Data Rumah Sakit yang Ditambah

5.2.15. Pengujian Fungsionalitas Mengedit Data Rumah Sakit

Tabel 5.16 Skenario 1 Pengujian Fungsionalitas Mengedit Data Rumah Sakit

No. Pengujian	**SCF-015**
Skenario Pengujian	Mengedit data rumah sakit dan disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data RS yang berisi data rumah sakit yang terdiri dari nama dan alamat rumah sakit
Aksi yang dilakukan	Menekan lama bagian data rumah sakit yang hendak diedit
Hasil yang diharapkan	Aplikasi dapat menampilkan data rumah sakit yang telah diedit pada halaman Kelola Data RS
Hasil yang diperoleh	Aplikasi menampilkan data rumah sakit yang telah diedit pada halaman Kelola Data RS
Hasil Pengujian	Berhasil

Langkah-langkah SCF-015:

1. Penguji masuk ke halaman Kelola Data RS dan menekan lama pada bagian data rumah sakit yang hendak diedit sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Edit Data RS" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan halaman Edit Data RS.
3. Penguji masuk ke halaman Edit Data RS dan mengedit nama dan alamat rumah sakit yang terdapat pada isian nama rumah sakit.
4. Penguji menyimpan data rumah sakit yang diedit dengan memilih tombol "Simpan". Aplikasi menampilkan semua data rumah sakit termasuk data

rumah sakit yang baru diedit pada halaman Kelola Data RS seperti pada Gambar 5.20.

Gambar 5.20 Halaman Kelola Data RS setelah Menyimpan Data Rumah Sakit yang Diedit

5.2.16. Pengujian Fungsionalitas Menghapus Data Rumah Sakit

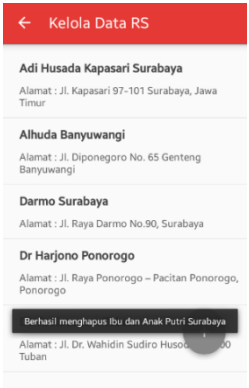
Tabel 5.17 Skenario 1 Pengujian Fungsionalitas Menghapus Data Rumah Sakit

Skenario Pengujian	**SCF-016**
Skenario Pengujian	Menghapus data rumah sakit dari sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data RS yang berisi data rumah sakit yang terdiri dari nama dan rumah sakit
Aksi yang dilakukan	Menekan lama bagian data rumah sakit yang hendak dihapus
Hasil yang diharapkan	Aplikasi dapat menampilkan *alert text* bahwa data rumah sakit yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data RS
Hasil yang diperoleh	Aplikasi menampilkan *alert text* bahwa data rumah sakit yang dipilih berhasil dihapus dan tidak terdapat

	pada halaman Kelola Data RS
Hasil Pengujian	Berhasil

Langkah-langkah SCF-016:

1. Penguji masuk ke halaman Kelola Data RS dan menekan lama pada bagian data rumah sakit yang hendak dihapus sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Hapus Data RS" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan *alert dialog* yang menunjukkan data rumah sakit yang dipilih berhasil dihapus dan halaman Kelola Data RS seperti pada Gambar 5.21.



Gambar 5.21 Halaman Kelola Data RS setelah Data Rumah Sakit Dihapus

5.2.17. Pengujian Fungsionalitas Melihat Data Jarak Penugasan

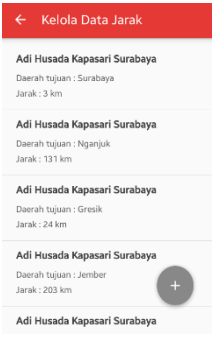
Tabel 5.18 Skenario 1 Pengujian Fungsionalitas Melihat Data Jarak Penugasan

No. Pengujian	**SCF-017**
Skenario Pengujian	Aplikasi menampilkan data jarak penugasan ketika admin masuk ke halaman Kelola Data Jarak
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data yang terdapat beberapa menu kelola data, yaitu "Daerah", "Bencana", "Keahlian", "Rumah Sakit", "Dokter", "Jarak"
Aksi yang dilakukan	Memilih tombol "Jarak"
Hasil yang diharapkan	Aplikasi dapat menampilkan data jarak penugasan beserta daerah dan rumah sakit dari masing-masing jarak penugasan
Hasil yang diperoleh	Aplikasi menampilkan data rumah sakit beserta daerah dan rumah sakit dari masing-masing jarak penugasan
Hasil Pengujian	Berhasil

Langkah-langkah SCF-017:

1. Penguji masuk ke halaman halaman Kelola Data seperti pada Gambar 5.1 dan mengisikan semua isian yang terdapat pada halaman tersebut.
2. Penguji menekan tombol yang bertuliskan "Jarak". Aplikasi menampilkan *progress dialog* sebagai tanda bahwa sistem sedang memproses pengambilan data jarak penugasan.
3. Aplikasi menampilkan data jarak penugasan beserta daerah dan rumah sakit dari masing-masing jarak penugasan seperti pada Gambar 5.22 yang

menunjukkan bahwa penguji telah sukses melihat data jarak penugasan.



Gambar 5.22 Halaman Kelola Data Jarak

5.2.18. Pengujian Fungsionalitas Menambah Data Jarak Penugasan

Tabel 5.19 Skenario 1 Pengujian Fungsionalitas Menambah Data Jarak Penugasan

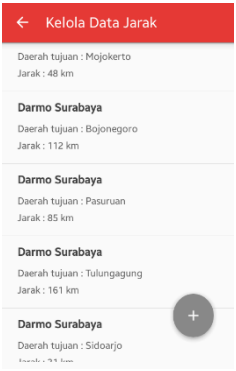
No. Pengujian	**SCF-018**
Skenario Pengujian	Menambah data jarak penugasan untuk disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Tambah Data Jarak yang berisikan isian jarak penugasan yang masih kosong, tombol yang bertuliskan “Pilih RS”, tombol yang bertuliskan “Pilih Daerah”, dan tombol yang bertuliskan “Simpan”
Aksi yang dilakukan	Mengisi isian jarak penugasan, memilih rumah sakit dan daerah pada halaman Tambah Data Jarak
Hasil yang diharapkan	Aplikasi dapat menampilkan data jarak penugasan yang telah ditambah pada halaman Kelola Data Jarak
Hasil yang diperoleh	Aplikasi menampilkan data jarak penugasan yang telah ditambah pada halaman Kelola Data Jarak
Hasil Pengujian	Berhasil

Langkah-langkah SCF-018:

1. Penguji masuk ke halaman Tambah Data Jarak seperti pada Gambar 5.23.

Gambar 5.23 Halaman Tambah Data Jarak

2. Penguji memilih rumah sakit dengan memilih tombol "Pilih RS" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar rumah sakit yang ada pada sistem.
3. Penguji memilih rumah sakit dengan memilih pilihan "OK" setelah memilih satu rumah sakit pada daftar rumah sakit. Aplikasi menampilkan daftar rumah sakit yang sudah dipilih.
4. Penguji memilih daerah dengan memilih tombol "Pilih Daerah" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar daerah yang ada pada sistem.
5. Penguji memilih daerah dengan memilih pilihan "OK" setelah memilih satu daerah pada daftar daerah. Aplikasi menampilkan daerah.
6. Mengisi isian jarak penugasan.
7. Penguji menyimpan data jarak penugasan yang ditambah dengan memilih tombol "Simpan". Aplikasi menampilkan semua data jarak penugasan termasuk data jarak penugasan yang baru ditambah pada halaman Kelola Data Jarak seperti pada Gambar 5.24.



Gambar 5.24 Halaman Kelola Data Jarak setelah Menyimpan Data Jarak yang Ditambah

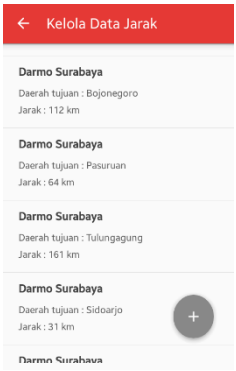
5.2.19. Pengujian Fungsionalitas Mengedit Data Jarak Penugasan

Tabel 5.20 Skenario 1 Pengujian Fungsionalitas Mengedit Data Jarak Penugasan

No. Pengujian	**SCF-019**
Skenario Pengujian	Mengedit data jarak penugasan dan disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Jarak yang berisi data jarak penugasan yang terdiri dari nama rumah sakit, nama daerah, dan jarak penugasan
Aksi yang dilakukan	Menekan lama bagian data jarak penugasan yang hendak diedit
Hasil yang diharapkan	Aplikasi dapat menampilkan data jarak penugasan yang telah diedit pada halaman Kelola Data Jarak
Hasil yang diperoleh	Aplikasi menampilkan data jarak penugasan yang telah diedit pada halaman Kelola Data Jarak
Hasil Pengujian	Berhasil

Langkah-langkah SCF-019:

1. Penguji masuk ke halaman Kelola Data Jarak dan menekan lama pada bagian data jarak penugasan yang hendak diedit sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Edit Data Jarak" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan halaman Edit Data Jarak.
3. Penguji masuk ke halaman Edit Data Jarak dan mengedit jarak penugasan yang terdapat pada isian jarak penugasan yang merepresentasikan jarak dari rumah sakit ke daerah yang terdapat pada halaman tersebut.
4. Penguji menyimpan data jarak penugasan yang diedit dengan memilih tombol "Simpan". Aplikasi menampilkan semua data jarak penugasan termasuk data jarak penugasan yang baru diedit pada halaman Kelola Data Jarak seperti pada Gambar 5.25.



Gambar 5.25 Halaman Kelola Data Jarak setelah Menyimpan Data Jarak Penugasan yang Diedit

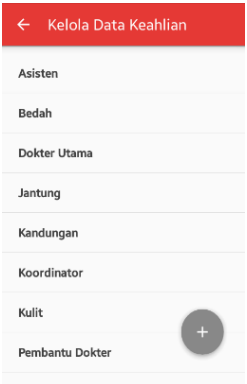
5.2.20. Pengujian Fungsionalitas Melihat Data Keahlian Dokter

Tabel 5.21 Skenario 1 Pengujian Fungsionalitas Melihat Data Keahlian Dokter

No. Pengujian	**SCF-020**
Skenario Pengujian	Aplikasi menampilkan data keahlian ketika admin masuk ke halaman Kelola Data Keahlian
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data yang terdapat beberapa menu kelola data, yaitu "Daerah", "Bencana", "Keahlian", "Rumah Sakit", "Dokter", "Jarak"
Aksi yang dilakukan	Memilih tombol "Keahlian"
Hasil yang diharapkan	Aplikasi dapat menampilkan data keahlian berupa nama keahlian
Hasil yang diperoleh	Aplikasi menampilkan data keahlian berupa nama keahlian
Hasil Pengujian	Berhasil

Langkah-langkah SCF-020:

1. Penguji masuk ke halaman halaman Kelola Data seperti pada Gambar 5.1.
2. Penguji menekan tombol yang bertuliskan "Keahlian". Aplikasi menampilkan *progress dialog* sebagai tanda bahwa sistem sedang memproses pengambilan data keahlian.
3. Aplikasi menampilkan data keahlian berupa nama keahlian seperti pada Gambar 5.26 yang menunjukkan bahwa penguji telah sukses melihat data keahlian.



Gambar 5.26 Halaman Kelola Data Keahlian

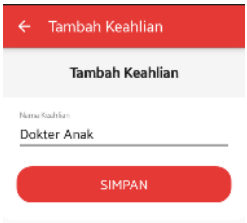
5.2.21. Pengujian Fungsionalitas Menambah Data Keahlian Dokter

Tabel 5.22 Skenario 1 Pengujian Fungsionalitas Menambah Data Keahlian Dokter

No. Pengujian	**SCF-021**
Skenario Pengujian	Menambah data keahlian untuk disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Tambah Data Keahlian yang berisikan isian nama keahlian yang masih kosong dan tombol yang bertuliskan “Simpan”
Aksi yang dilakukan	Mengisi isian nama keahlian pada halaman Tambah Data Keahlian
Hasil yang diharapkan	Aplikasi dapat menampilkan data keahlian yang telah ditambah pada halaman Kelola Data Keahlian
Hasil yang diperoleh	Aplikasi menampilkan data keahlian yang telah ditambah pada halaman Kelola Data Keahlian
Hasil Pengujian	Berhasil

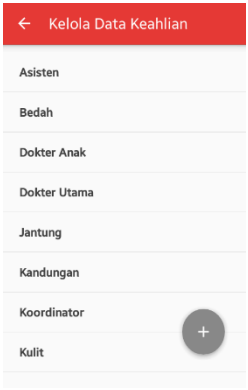
Langkah-langkah SCF-021:

1. Penguji masuk ke halaman Tambah Data Keahlian seperti pada Gambar 5.27 dan mengisi isian nama keahlian.



Gambar 5.27 Halaman Tambah Data Keahlian ketika Mengisi Nama Keahlian

2. Penguji menyimpan data keahlian yang ditambah dengan memilih tombol "Simpan". Aplikasi menampilkan semua data keahlian termasuk data keahlian yang baru ditambah pada halaman Kelola Data Keahlian seperti pada Gambar 5.28.



Gambar 5.28 Halaman Kelola Data Keahlian setelah Menyimpan Data Keahlian yang Ditambah

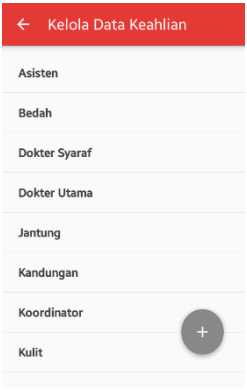
5.2.22. Pengujian Fungsionalitas Mengedit Data Keahlian Dokter

Tabel 5.23 Skenario 1 Pengujian Fungsionalitas Mengedit Data Keahlian Dokter

No. Pengujian	**SCF-022**
Skenario Pengujian	Mengedit data keahlian dan disimpan pada sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Keahlian yang berisi data keahlian yang terdiri dari nama keahlian
Aksi yang dilakukan	Menekan lama bagian data keahlian yang hendak diedit
Hasil yang diharapkan	Aplikasi dapat menampilkan data keahlian yang telah diedit pada halaman Kelola Data Keahlian
Hasil yang diperoleh	Aplikasi menampilkan data keahlian yang telah diedit pada halaman Kelola Data Keahlian
Hasil Pengujian	Berhasil

Langkah-langkah SCF-022:

1. Penguji masuk ke halaman Kelola Data Keahlian dan menekan lama pada bagian data rumah sakit yang hendak diedit sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Edit Data Keahlian" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan halaman Edit Data Keahlian.
3. Penguji masuk ke halaman Edit Data Keahlian dan mengedit nama keahlian yang terdapat pada isian nama keahlian.
4. Penguji menyimpan data keahlian yang diedit dengan memilih tombol "Simpan". Aplikasi menampilkan semua data keahlian termasuk data keahlian yang baru diedit pada halaman Kelola Data Keahlian seperti pada Gambar 5.29.



Gambar 5.29 Halaman Kelola Data Keahlian setelah Menyimpan Data Keahlian Dokter yang Diedit

5.2.23. Pengujian Fungsionalitas Menghapus Data Keahlian Dokter

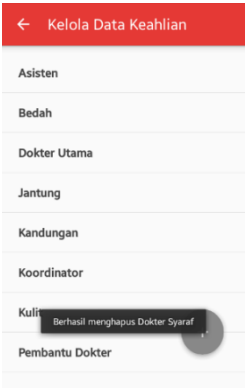
Tabel 5.24 Skenario 1 Pengujian Fungsionalitas Menghapus Data Keahlian Dokter

No. Pengujian	**SCF-023**
Skenario Pengujian	Menghapus data keahlian dari sistem
Kondisi Awal	Aplikasi menampilkan halaman Kelola Data Keahlian yang berisi data keahlian yang terdiri dari nama keahlian
Aksi yang dilakukan	Menekan lama bagian data keahlian yang hendak dihapus
Hasil yang diharapkan	Aplikasi dapat menampilkan *alert text* bahwa data keahlian yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Keahlian
Hasil yang diperoleh	Aplikasi menampilkan *alert text* bahwa data keahlian yang dipilih berhasil dihapus dan tidak terdapat pada halaman Kelola Data Keahlian
Hasil	Berhasil

Pengujian	

Langkah-langkah SCF-023:

1. Penguji masuk ke halaman Kelola Data Keahlian dan menekan lama pada bagian data keahlian yang hendak dihapus sehingga muncul *alert dialog* "Pilih Aksi".
2. Penguji memilih pilihan "Hapus Data Keahlian" pada *alert dialog* "Pilih Aksi". Aplikasi menampilkan *alert dialog* yang menunjukkan data keahlian yang dipilih berhasil dihapus dan halaman Kelola Data Keahlian seperti pada Gambar 5.30.



Gambar 5.30 Halaman Kelola Data Keahlian setelah Data Keahlian Dokter Dihapus

5.2.24. Pengujian Fungsionalitas Mengatur Penugasan Dokter

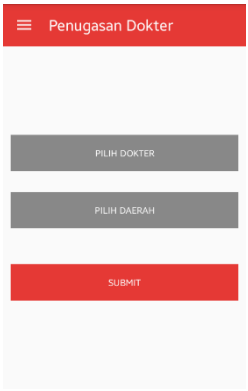
Tabel 5.25 Skenario 1 Pengujian Fungsionalitas Mengatur Penugasan Dokter

No. Pengujian	**SCF-024**
Skenario Pengujian	Mengatur penugasan dokter

Kondisi Awal	Aplikasi menampilkan halaman Penugasan yang berisi tombol yang bertuliskan "Pilih Dokter", tombol yang bertuliskan "Pilih Daerah", dan tombol yang bertuliskan "Submit"
Aksi yang dilakukan	Memilih dokter dan daerah pada halaman Penugasan, serta memilih tombol "Simpan" pada halaman Penugasan Hasil
Hasil yang diharapkan	Aplikasi dapat menampilkan hasil penugasan berupa dokter yang ditugaskan pada daerah yang sudah dipilih sebelumnya, serta jarak masing-masing penugasan pada halaman Penugasan Hasil dan menyimpannya pada sistem
Hasil yang diperoleh	Aplikasi menampilkan hasil penugasan berupa dokter yang ditugaskan pada daerah yang sudah dipilih sebelumnya, serta jarak masing-masing penugasan pada halaman Penugasan Hasil dan menyimpannya pada sistem
Hasil Pengujian	Berhasil

Langkah-langkah SCF-024:

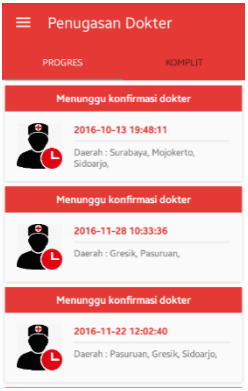
1. Penguji masuk ke halaman Penugasan seperti pada Gambar 5.31.



Gambar 5.31 Halaman Penugasan Dokter

2. Penguji memilih dokter pengganti dengan memilih tombol "Pilih Dokter" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar dokter yang ada pada sistem.
3. Penguji memilih dokter pengganti dengan memilih pilihan "OK" setelah memilih beberapa calon dokter pengganti pada daftar dokter. Aplikasi menampilkan daftar dokter yang sudah dipilih.
4. Penguji memilih daerah dengan memilih tombol "Pilih Daerah" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar daerah yang ada pada sistem.
5. Penguji memilih daerah dengan memilih pilihan "OK" setelah memilih beberapa daerah pada daftar daerah. Aplikasi menampilkan daerah yang sudah dipilih.
6. Penguji memilih tombol "Submit". Aplikasi menampilkan semua data hasil penugasan yang terdiri dari nama dokter, nama daerah, dan jarak dari masing-masing hasil penugasan pada halaman Penugasan.
7. Penguji menyimpan data hasil penugasan dengan memilih tombol "Simpan". Aplikasi menampilkan halaman Riwayat Progres yang menunjukkan hasil

penugasan berhasil disimpan seperti pada Gambar 5.32.



Gambar 5.32 Halaman Riwayat Progres setelah Menyimpan Hasil Penugasan

5.2.25. Pengujian Fungsionalitas Mengatur Penugasan Ulang Dokter

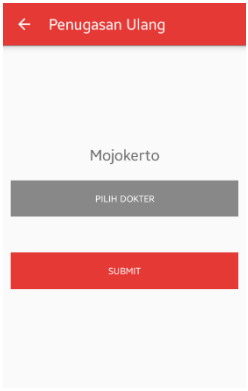
Tabel 5.26 Skenario 1 Pengujian Fungsionalitas Mengatur Penugasan Ulang Dokter

No. Pengujian	**SCF-025**
Skenario Pengujian	Mengatur penugasan ulang dokter
Kondisi Awal	Aplikasi menampilkan halaman Penugasan Ulang yang berisi nama daerah, tombol yang bertuliskan "Pilih Dokter" dan tombol yang bertuliskan "Submit"
Aksi yang dilakukan	Memilih dokter pada halaman Penugasan Ulang dan memilih tombol "Simpan" pada halaman Penugasan Ulang Hasil
Hasil yang diharapkan	Aplikasi dapat menampilkan hasil penugasan ulang berupa dokter yang ditugaskan pada daerah yang sudah dipilih sebelumnya sebagai dokter pengganti,

	serta jarak masing-masing penugasan pada halaman Penugasan Ulang Hasil dan menyimpannya pada sistem
Hasil yang diperoleh	Aplikasi menampilkan hasil penugasan ulang berupa dokter yang ditugaskan pada daerah yang sudah dipilih sebelumnya sebagai dokter pengganti, serta jarak masing-masing penugasan pada halaman Penugasan Ulang Hasil dan menyimpannya pada sistem
Hasil Pengujian	Berhasil

Langkah-langkah SCF-025:

1. Penguji masuk ke halaman Penugasan Ulang seperti pada Gambar 5.33.

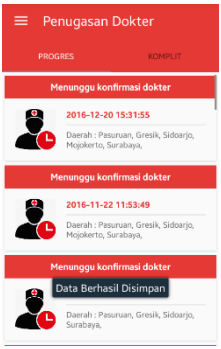


Gambar 5.33 Halaman Penugasan Ulang Dokter

2. Penguji memilih dokter pengganti dengan memilih tombol "Pilih Dokter" terlebih dahulu. Aplikasi menampilkan *alert dialog* yang menunjukkan daftar dokter yang ada pada sistem.
3. Penguji memilih dokter pengganti dengan memilih pilihan "OK" setelah memilih beberapa calon dokter

pengganti pada daftar dokter. Aplikasi menampilkan daftar dokter yang sudah dipilih.

4. Penguji memilih tombol "Submit". Aplikasi menampilkan semua data hasil penugasan yang terdiri dari nama dokter, nama daerah, dan jarak dari masing-masing hasil penugasan pada halaman Penugasan.
5. Penguji menyimpan data hasil penugasan dengan memilih tombol "Simpan". Aplikasi menampilkan *alert text* "Data Berhasil Disimpan" yang menunjukkan hasil penugasan berhasil disimpan pada halaman Riwayat Progres seperti pada Gambar 5.34.



Gambar 5.34 Halaman Riwayat Progres setelah Menyimpan Hasil Penugasan Ulang

5.2.26. Pengujian Fungsionalitas Melihat Daftar Penugasan

Tabel 5.27 Skenario 1 Pengujian Fungsionalitas Melihat Daftar Penugasan

No. Pengujian	**SCF-026**
Skenario Pengujian	Aplikasi menampilkan daftar penugasan ketika dokter masuk ke halaman Daftar Penugasan
Kondisi Awal	Aplikasi menampilkan halaman Menu Utama yang terdapat tombol "Daftar Penugasan"

Aksi yang dilakukan	Memilih tombol “Daftar Penugasan”
Hasil yang diharapkan	Aplikasi dapat menampilkan daftar penugasan berupa waktu penugasan, nama daerah yang membutuhkan penugasan, dan jarak
Hasil yang diperoleh	Aplikasi menampilkan daftar penugasan berupa nama daerah yang membutuhkan penugasan, jarak, dan waktu penugasan.
Hasil Pengujian	Berhasil

Langkah-langkah SCF-026:

1. Penguji masuk ke halaman halaman Menu Utama seperti pada Gambar 5.35.

Gambar 5.35 Halaman Menu Utama

2. Penguji menekan tombol yang bertuliskan “Daftar Keahlian”. Aplikasi menampilkan *progress dialog* sebagai tanda bahwa sistem sedang memproses pengambilan data daftar penugasan.
3. Aplikasi menampilkan data daftar penugasan berupa nama daerah yang membutuhkan penugasan, jarak, dan waktu penugasan seperti pada Gambar 5.36 yang menunjukkan bahwa penguji telah sukses melihat data daftar penugasan.

Gambar 5.36 Halaman Daftar Penugasan

5.2.27. Pengujian Fungsionalitas Menyetujui Penugasan

Tabel 5.28 Skenario 1 Pengujian Fungsionalitas Menyetujui Penugasan

No. Pengujian	**SCF-027**
Skenario Pengujian	Aplikasi memproses konfirmasi persetujuan dokter terhadap suatu penugasan
Kondisi Awal	Aplikasi menampilkan halaman Daftar Penugasan yang terdiri dari nama daerah yang membutuhkan penugasan, jarak, dan waktu penugasan pada setiap daftar penugasan
Aksi yang dilakukan	Memilih salah satu daftar penugasan yang hendak disetujui dan memilih pilihan "Iya" pada *alert dialog* "Konfirmasi Penugasan"
Hasil yang diharapkan	Aplikasi dapat menyimpan persetujuan dokter untuk mengambil penugasan yang dipilih
Hasil yang diperoleh	Aplikasi menyimpan persetujuan dokter untuk mengambil penugasan yang dipilih
Hasil Pengujian	Berhasil

Langkah-langkah SCF-027:

1. Penguji masuk ke halaman halaman Daftar Penugasan seperti pada Gambar 5.37.

Gambar 5.37 Halaman Daftar Penugasan

2. Penguji memilih salah satu penugasan pada halaman Daftar Penugasan. Aplikasi menampilkan *alert dialog* "Konfirmasi Penugasan".
3. Penguji memilih pilihan "Iya". Aplikasi menampilkan halaman Aktivitas seperti pada Gambar 5.38 yang berguna sebagai konfirmasi dokter jika dokter telah menyelesaikan penugasannya.

Gambar 5.38 Halaman Aktivitas setelah Penugasan Disetujui

5.2.28. Pengujian Fungsionalitas Menolak Penugasan

Tabel 5.29 Skenario 1 Pengujian Fungsionalitas Menolak Penugasan

No.	**SCF-028**

Pengujian	
Skenario Pengujian	Aplikasi memproses konfirmasi penolakan dokter terhadap suatu penugasan
Kondisi Awal	Aplikasi menampilkan halaman Daftar Penugasan yang terdiri dari nama daerah yang membutuhkan penugasan, jarak, dan waktu penugasan pada setiap daftar penugasan
Aksi yang dilakukan	Memilih salah satu daftar penugasan yang hendak disetujui dan memilih pilihan "Tidak" pada *alert dialog* "Konfirmasi Penugasan"
Hasil yang diharapkan	Aplikasi dapat menyimpan penolakan dokter untuk mengambil penugasan yang dipilih
Hasil yang diperoleh	Aplikasi menyimpan penolakan dokter untuk mengambil penugasan yang dipilih
Hasil Pengujian	Berhasil

Langkah-langkah SCF-028:

1. Penguji masuk ke halaman halaman Daftar Penugasan seperti pada Gambar 5.39.

Gambar 5.39 Halaman Daftar Penugasan

2. Penguji memilih salah satu penugasan pada halaman Daftar Penugasan. Aplikasi menampilkan *alert dialog* "Konfirmasi Penugasan".
3. Penguji memilih pilihan "Tidak". Aplikasi menampilkan *alert text* bahwa penugasan ditolak pada halaman Daftar Penugasan seperti pada Gambar 5.40.

Gambar 5.40 Halaman Daftar Penugasan ketika Penugasan Ditolak

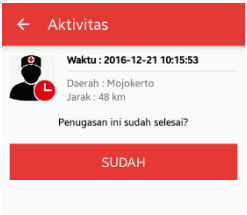
5.2.29. Pengujian Fungsionalitas Memberikan Konfirmasi Penugasan Selesai

Tabel 5.30 Skenario 1 Pengujian Fungsionalitas Memberikan Konfirmasi Penugasan Selesai

No. Pengujian	**SCF-029**
Skenario Pengujian	Aplikasi memproses konfirmasi dokter bahwa penugasan selesai
Kondisi Awal	Aplikasi menampilkan halaman Aktivitas yang terdiri dari waktu penugasan, nama daerah bencana tempat dokter ditugaskan, jarak penugasan, pertanyaan "Apakah Anda yakin telah menyelesaikan Penugasan ini?", serta tombol yang bertuliskan "Sudah"
Aksi yang dilakukan	Memilih tombol yang bertuliskan "Sudah" pada halaman Aktivitas sesuai dengan data penugasan yang telah dipilih dari halaman Riwayat Progres
Hasil yang diharapkan	Aplikasi dapat menyimpan konfirmasi dokter bahwa penugasan selesai
Hasil yang diperoleh	Aplikasi menyimpan konfirmasi dokter bahwa penugasan selesai
Hasil Pengujian	Berhasil

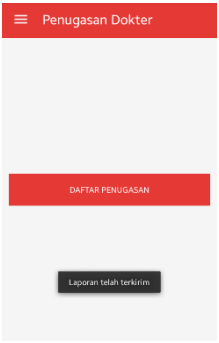
Langkah-langkah SCF-029:

1. Penguji masuk ke halaman halaman Riwayat Progres seperti pada Gambar 5.41.



Gambar 5.41 Halaman Riwayat Progres

2. Penguji memilih salah satu penugasan yang sudah selesai pada halaman Riwayat Progres. Aplikasi menampilkan halaman Aktivitas.
3. Penguji memilih pilihan "Sudah". Pada halaman Aktivitas, aplikasi menampilkan *alert text* bahwa konfirmasi penugasan selesai telah berhasil diproses dan disimpan, seperti pada Gambar 5.42.



Gambar 5.42 Halaman Daftar Penugasan ketika Penugasan Selesai Berhasil Diproses dan Disimpan

Gambar 5.43 Ilustrasi Cara Perolehan Data Uji Jarak Penugasan

5.3 Evaluasi Pengujian

Berdasarkan pengujian yang telah dilakukan, semua pengujian fungsionalitas memberikan hasil yang sesuai dengan skenario yang direncanakan. Evaluasi terhadap pengujian yang dilaksanakan, baik pengujian fungsionalitas dijelaskan sebagai berikut:

1. Fungsionalitas melihat, menambah, mengedit, dan menghapus data daerah berjalan sesuai yang diharapkan. Pengujian SCF-001, SCF-002, SCF-003, dan SCF-004 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan masing-masing skenario pengujian.
2. Fungsionalitas melihat, menambah, mengedit, dan menghapus data bencana berjalan sesuai yang diharapkan. Pengujian SCF-005, SCF-006, SCF-007, dan SCF-008 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan skenario pengujian.
3. Fungsionalitas melihat, menambah, mengedit, dan menghapus data dokter berjalan sesuai yang diharapkan. Pengujian SCF-009, SCF-010, SCF-011, dan SCF-012 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan masing-masing skenario pengujian.
4. Fungsionalitas melihat, menambah, mengedit, dan menghapus data rumah sakit berjalan sesuai yang diharapkan. Pengujian SCF-013, SCF-014, SCF-015, dan

SCF-016 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan skenario pengujian.

5. Fungsionalitas melihat, menambah, dan mengedit data jarak penugasan berjalan sesuai yang diharapkan. Pengujian SCF-017, SCF-018, dan SCF-019 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan skenario pengujian.
6. Fungsionalitas melihat, menambah, mengedit, dan menghapus data keahlian dokter berjalan sesuai yang diharapkan. Pengujian SCF-020, SCF-021, SCF-022, dan SCF-023 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan skenario pengujian.
7. Fungsionalitas mengatur penugasan dokter dan penugasan ulang dokter berjalan sesuai yang diharapkan. Pengujian SCF-024 dan SCF-025 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan skenario pengujian.
8. Fungsionalitas melihat daftar penugasan, menyetujui penugasan, menolak penugasan, dan memberikan konfirmasi penugasan selesai berjalan sesuai yang diharapkan. Pengujian SCF-026, SCF-027, SCF-028, dan SCF-029 memperlihatkan bahwa sistem memberikan respon yang sesuai dengan skenario pengujian.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil pengujian yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk pengembangan perangkat lunak di masa mendatang.

6.1 Kesimpulan

1. Sistem penjadwalan/penugasan dokter pada keadaan darurat (bencana) dengan metode Graf *Bipartite*, Algoritma *Ford-Fulkerson*, dan *Integer Programming* dengan Metode Penugasan Optimal untuk *Minimization Case* dapat diakses melalui perangkat *mobile* ini.
2. Uji coba aplikasi perangkat lunak ini dilakukan pada studi kerawanan bencana, khususnya di wilayah Jawa Timur dengan menggunakan data-data *dummy* pada *database*.

6.2 Saran

1. Penambahan fungsional untuk menambah *user*, sehingga *user* dengan hak akses admin dapat ditambah tanpa harus mengakses *database* secara langsung.
2. Penggunaan data *real* pada *database* sebagai uji coba aplikasi perangkat lunak ini.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

[1] Wikipedia, "Android (operating system)," 2016. [Online]. Available: https://id.wikipedia.org/wiki/Android_(sistem_operasi).

[2] A. F. Rachman, "Android Kuasai Asia Tenggara, di Indonesia Paling Juara," inet.detik.com, 27 10 2015. [Online]. Available: http://inet.detik.com/read/2015/10/27/103538/3054169/317/android-kuasai-asia-tenggara-di-indonesia-paling-juara.

[3] C. Pautasso, REST: Advanced Research Topics and Practical Applications, Springer, 2014.

[4] MySQL, "MySQL. The World's Most Popular Open Source Database," [Online]. Available: https://mysql.com. [Accessed 16 Januari 2017].

[5] A. Developers, "Transmitting Network Data Using Volley," [Online]. Available: https://developer.android.com/training/volley/index.html.

[6] A. Developers, "Sending a Simple Request," [Online]. Available: https://developer.android.com/training/volley/simple.html. [Accessed 29 September 2016].

[7] A. Developers, "Making a Standard Request," [Online]. Available: https://developer.android.com/training/volley/request.html. [Accessed 29 September 2016].

[8] D. Suhardjo, "Arti Penting Pendidikan Mitigasi Bencana Dalam Mengurangi Resiko Bencana," *Cakrawala Pendidikan*, vol. 2, 2011.

[9] Wikipedia, "Persiapan Bencana," 28 Mei 2016. [Online]. Available:

https://id.wikipedia.org/wiki/Persiapan_bencana. [Accessed 29 September 2016].

[10] Z. Fathoni, "Algoritma Penentuan Graf Bipartit," [Online]. Available: http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2009-2010/Makalah0910/MakalahStrukdis0910-105.pdf. [Accessed 28 September 2016].

[11] L. R. a. D. R. F. Ford Jr, Flows in networks, Princeton University Press, 2015.

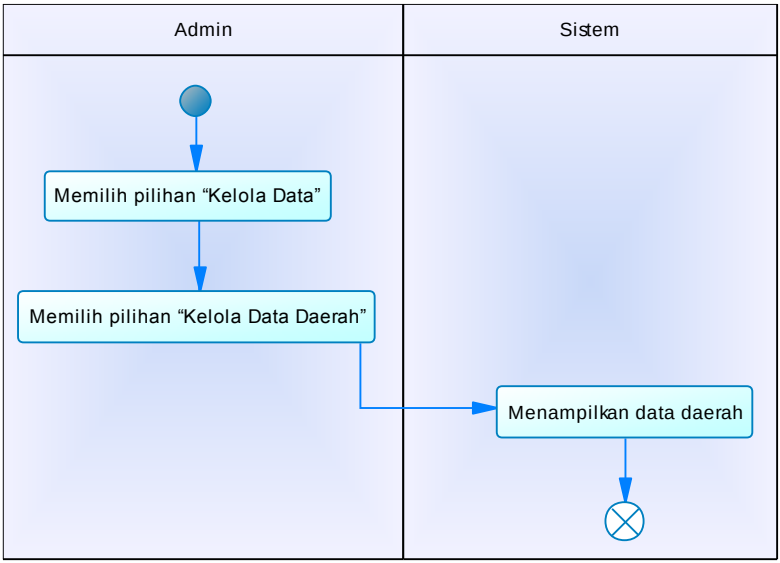
[12] M. M. S. Gaglani, "A Study on Transportation Problem, Transshipment Problem, Assignment Problem and Supply Chain Management," *Department of Statistics, Saurashtra University,* pp. 80-97, 2011.

[13] "Google Maps," Google, [Online]. Available: https://maps.google.com. [Accessed 14 Januari 2017].

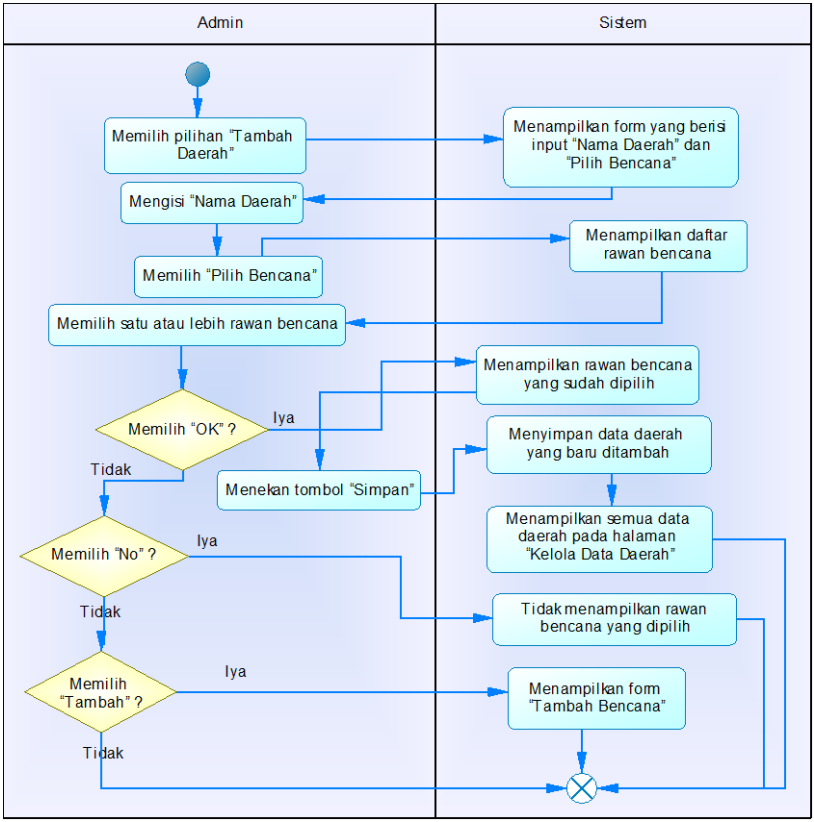
[14] Techinasia, "Android mendominasi smartphone di Indonesia," 22 Desember 2015. [Online]. Available: https://id.techinasia.com/android-opera-dominasi-smartphone-indonesia-2014/.

[15] "Daftar Rumah Sakit," [Online]. Available: http://rumah-sakit.findthebest.co.id. [Accessed 14 Januari 2017].

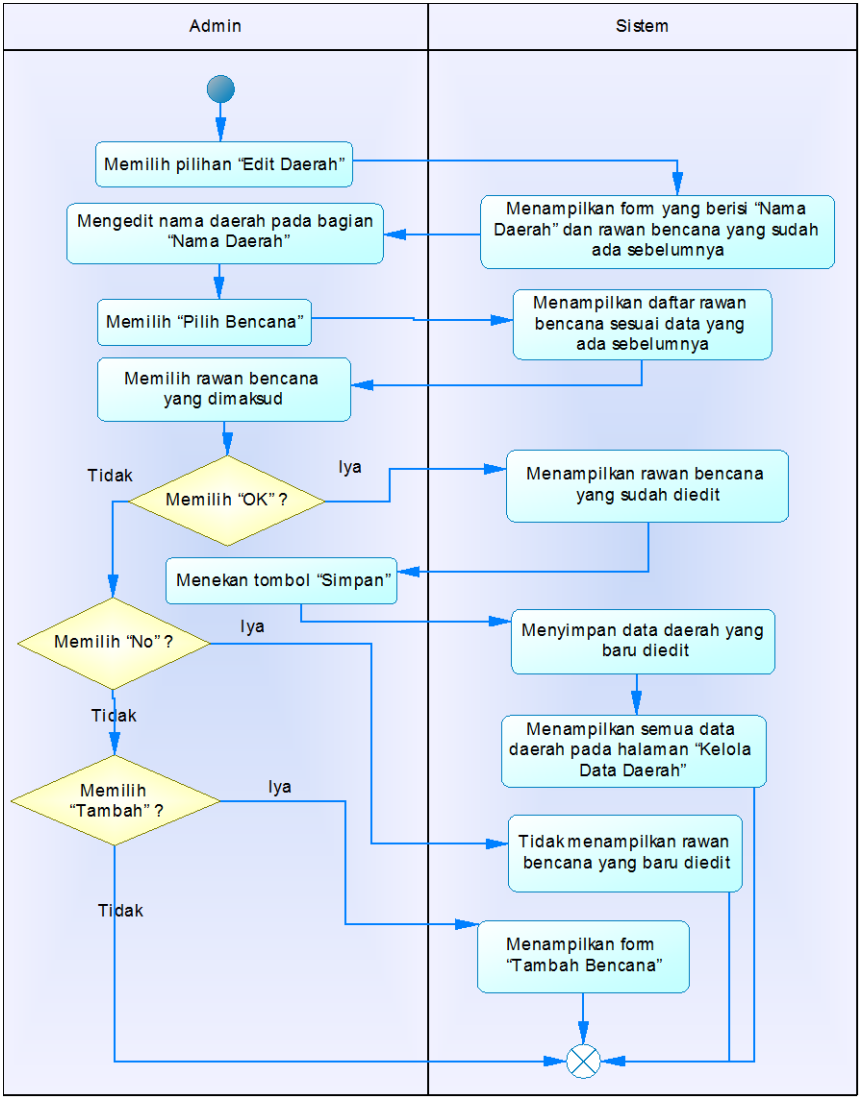
LAMPIRAN



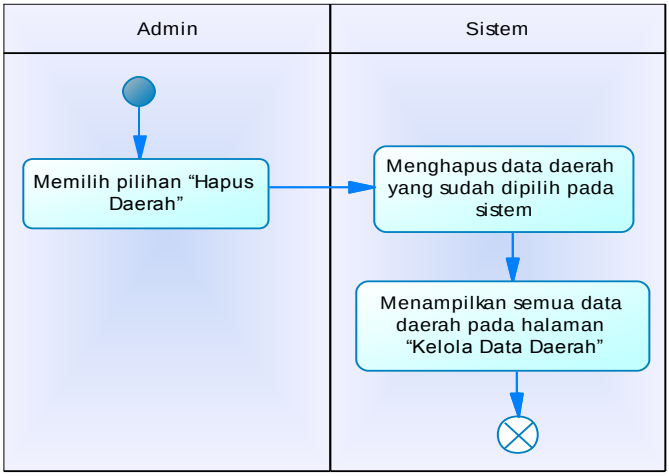
Gambar 0.1 Diagram Aktivitas UC-0001



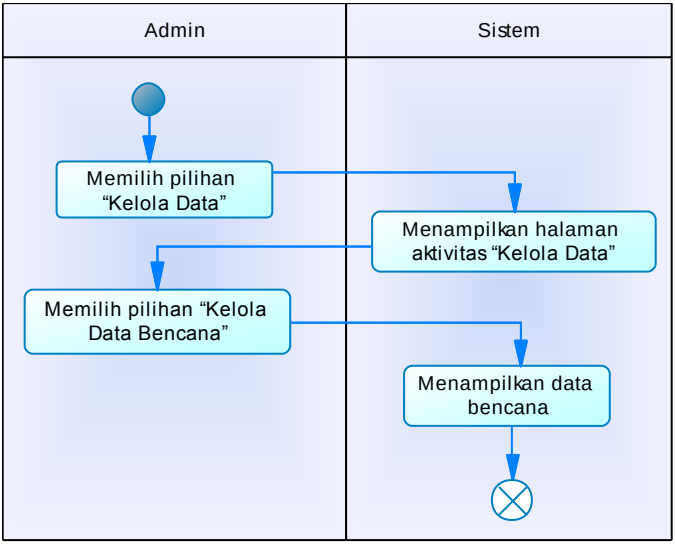
Gambar 0.2 Diagram Aktivitas UC-0002



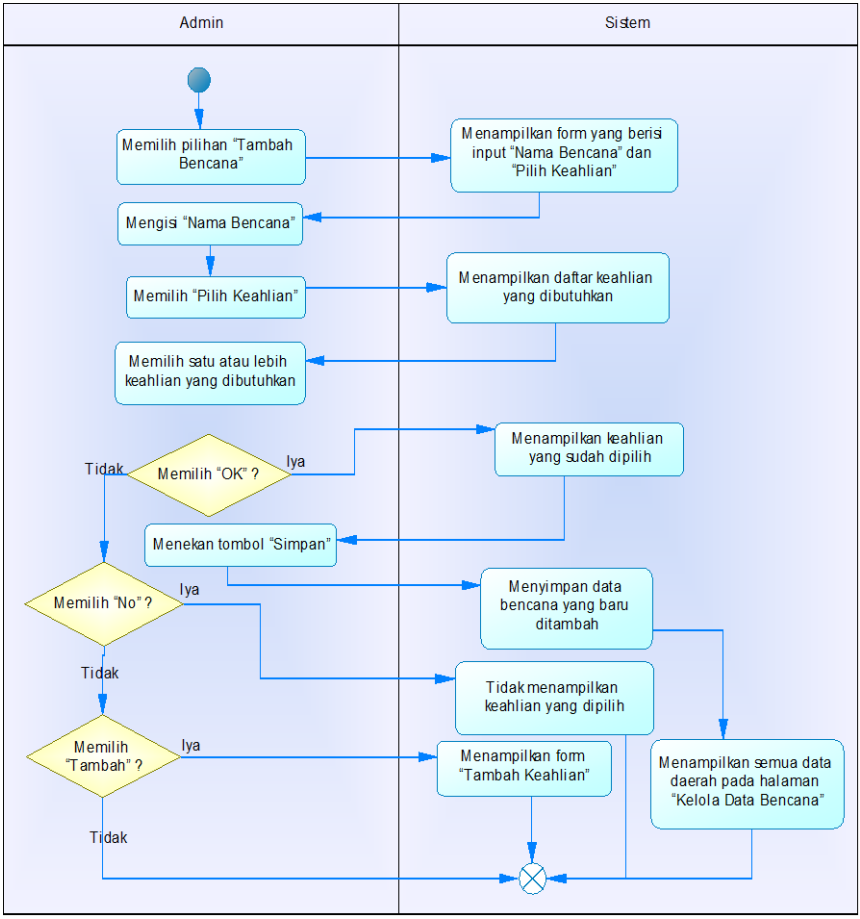
Gambar 0.3 Diagram Aktivitas UC-0003



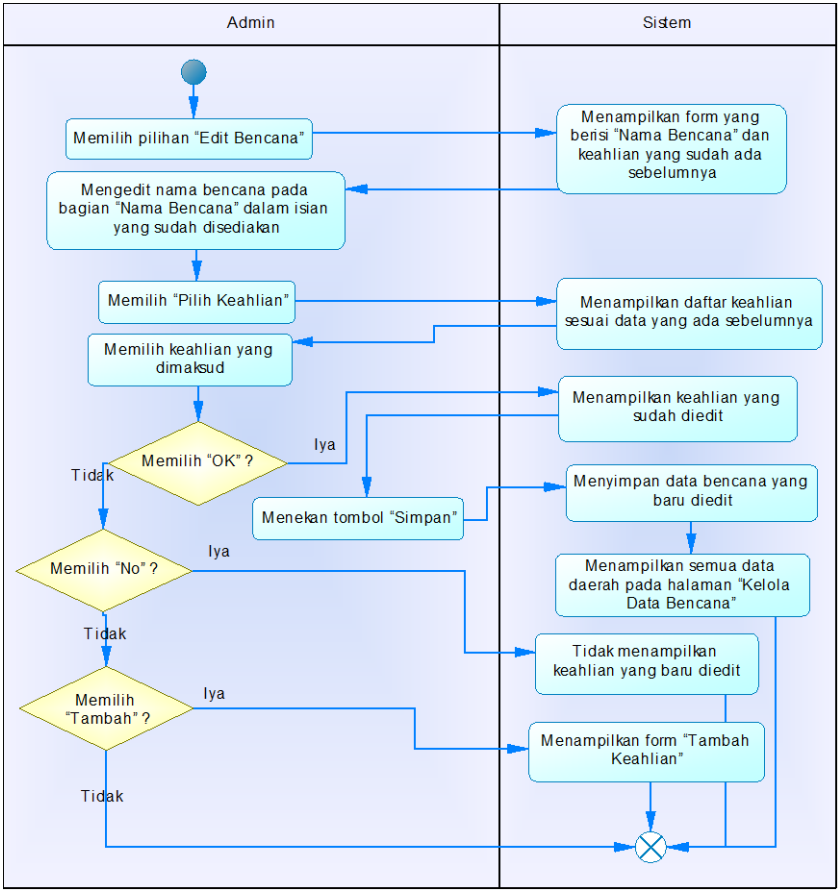
Gambar 0.4 Diagram Aktivitas UC-0004



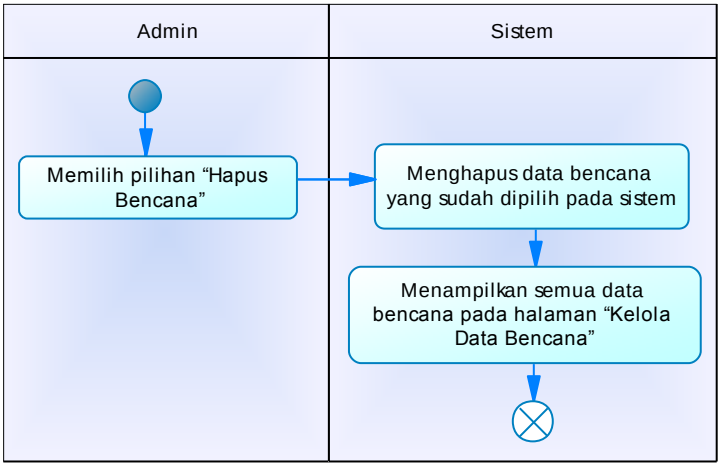
Gambar 0.5 Diagram Aktivitas UC-0005



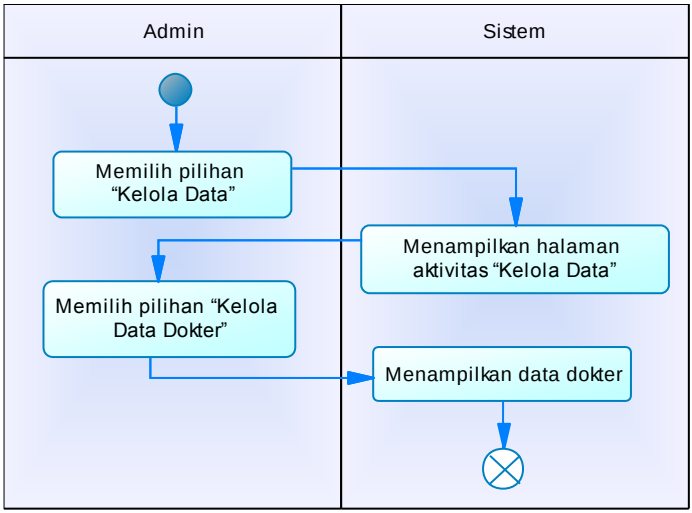
Gambar 0.6 Diagram Aktivitas UC-0006



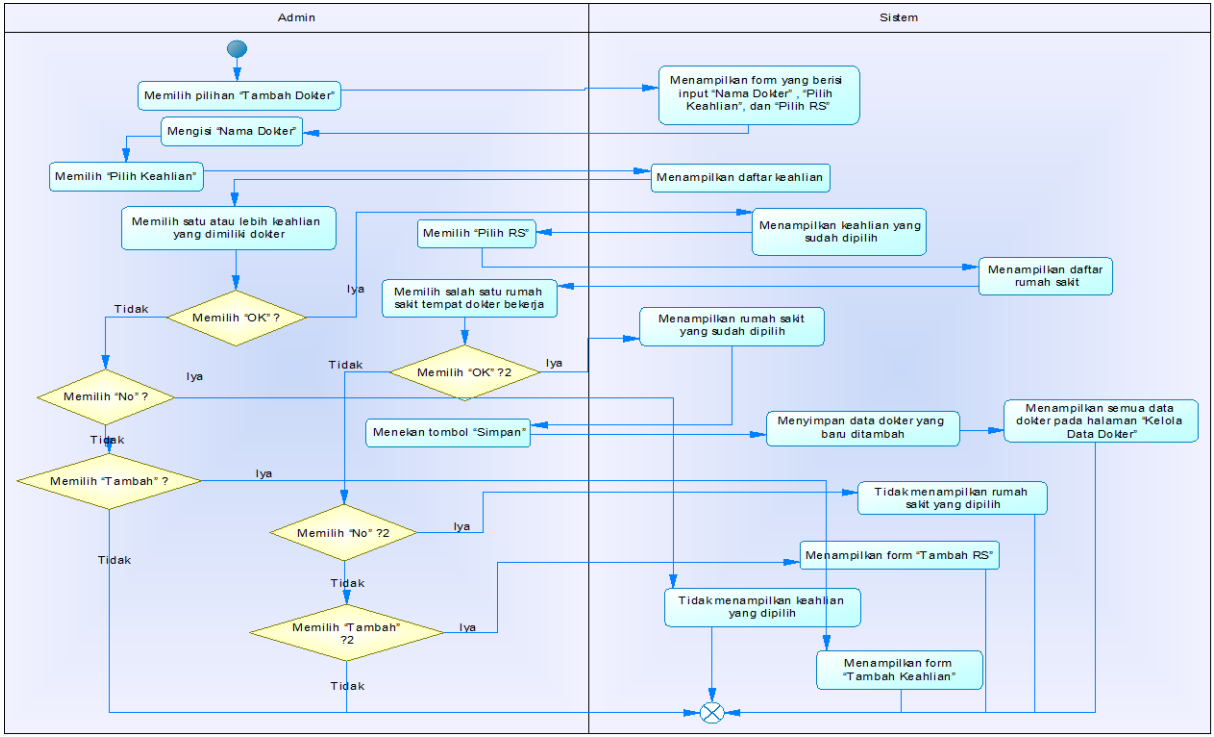
Gambar 0.7 Diagram Aktivitas UC-0007



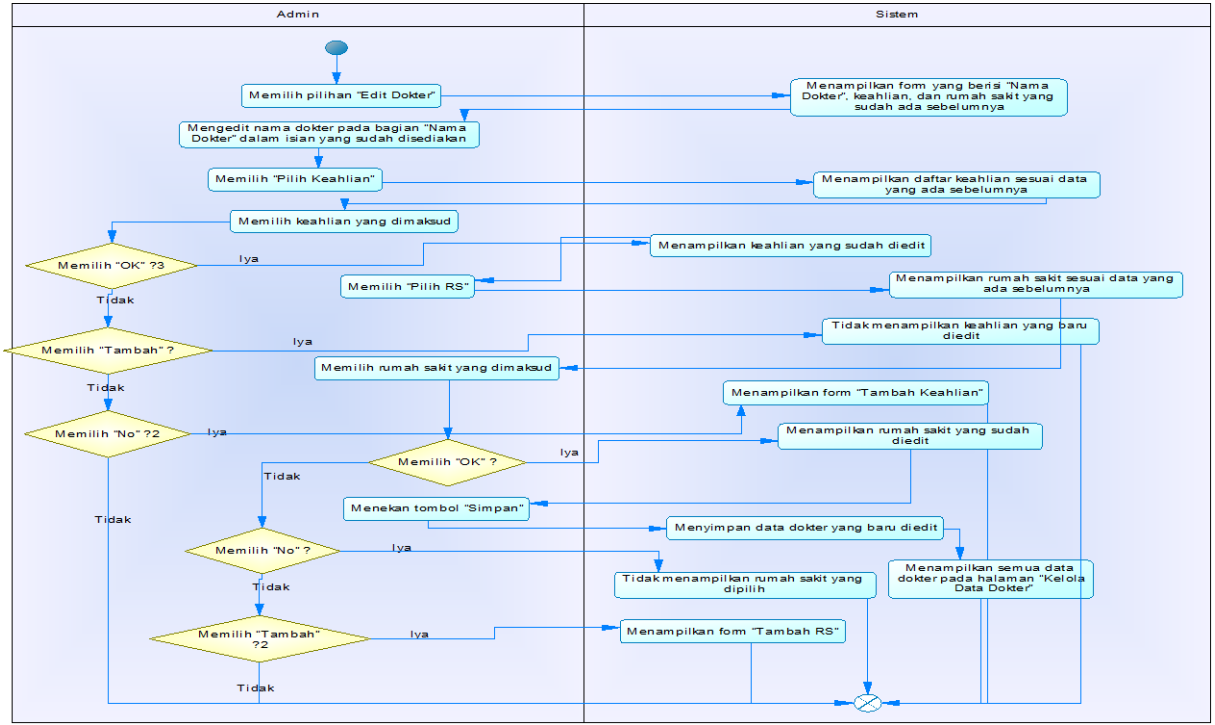
Gambar 0.8 Diagram Aktivitas UC-0008



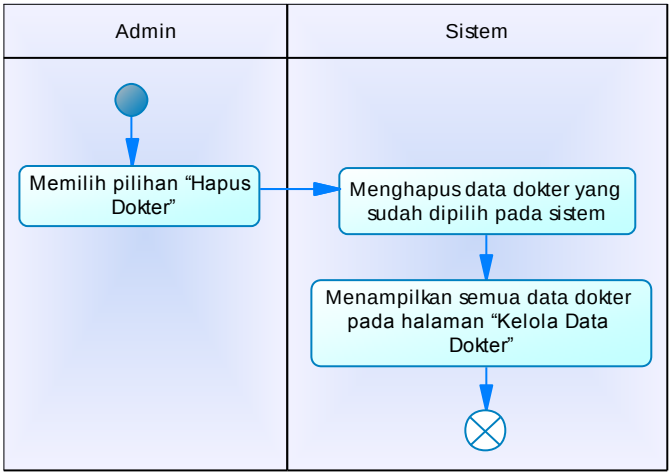
Gambar 0.9 Diagram Aktivitas UC-0009



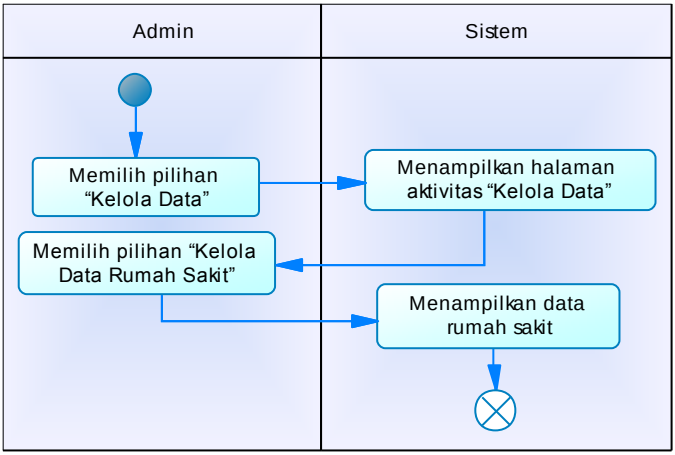
Gambar 0.10 Diagram Aktivitas UC-0010



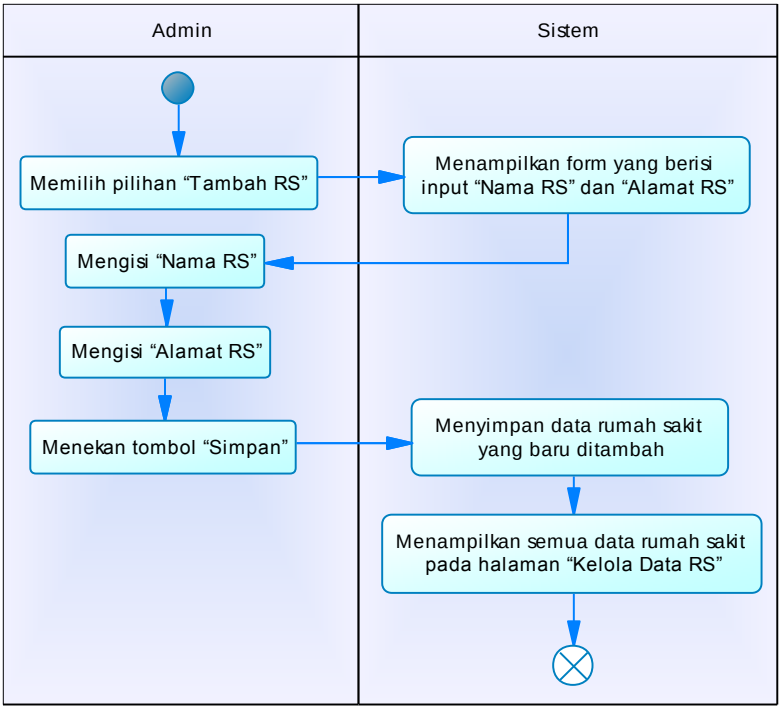
Gambar 0.11 Diagram Aktivitas UC-0011



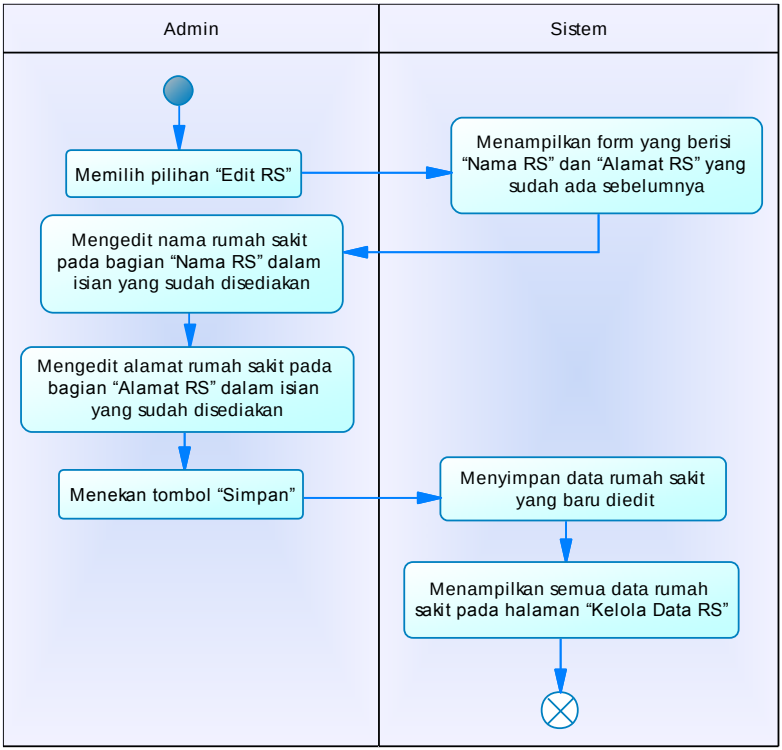
Gambar 0.12 Diagram Aktivitas UC-0012



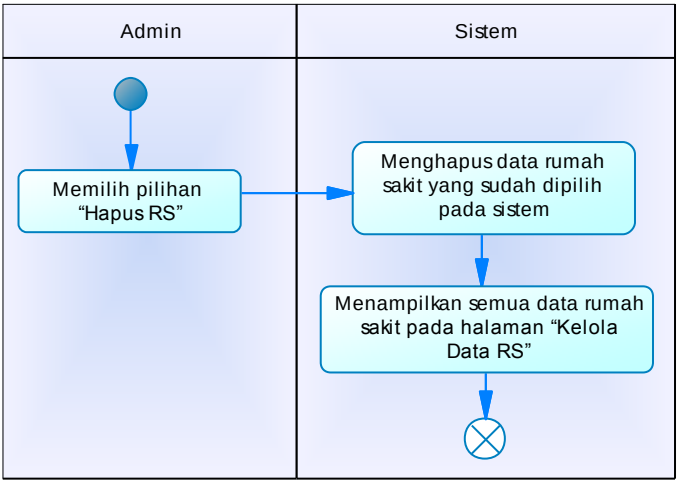
Gambar 0.13 Diagram Aktivitas UC-0013



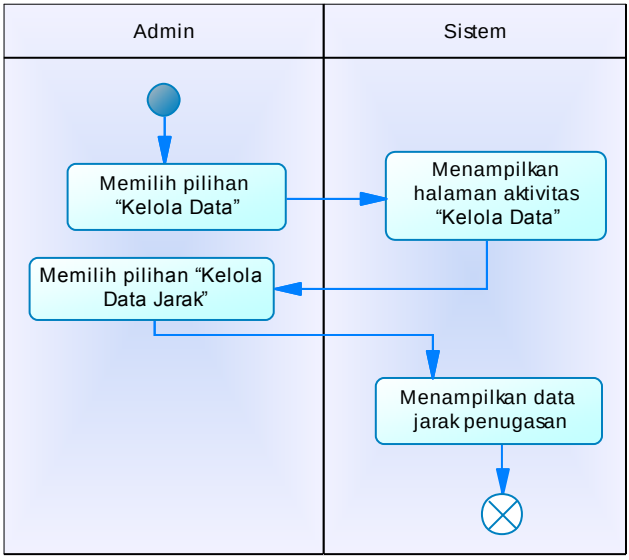
Gambar 0.14 Diagram Aktivitas UC-0014



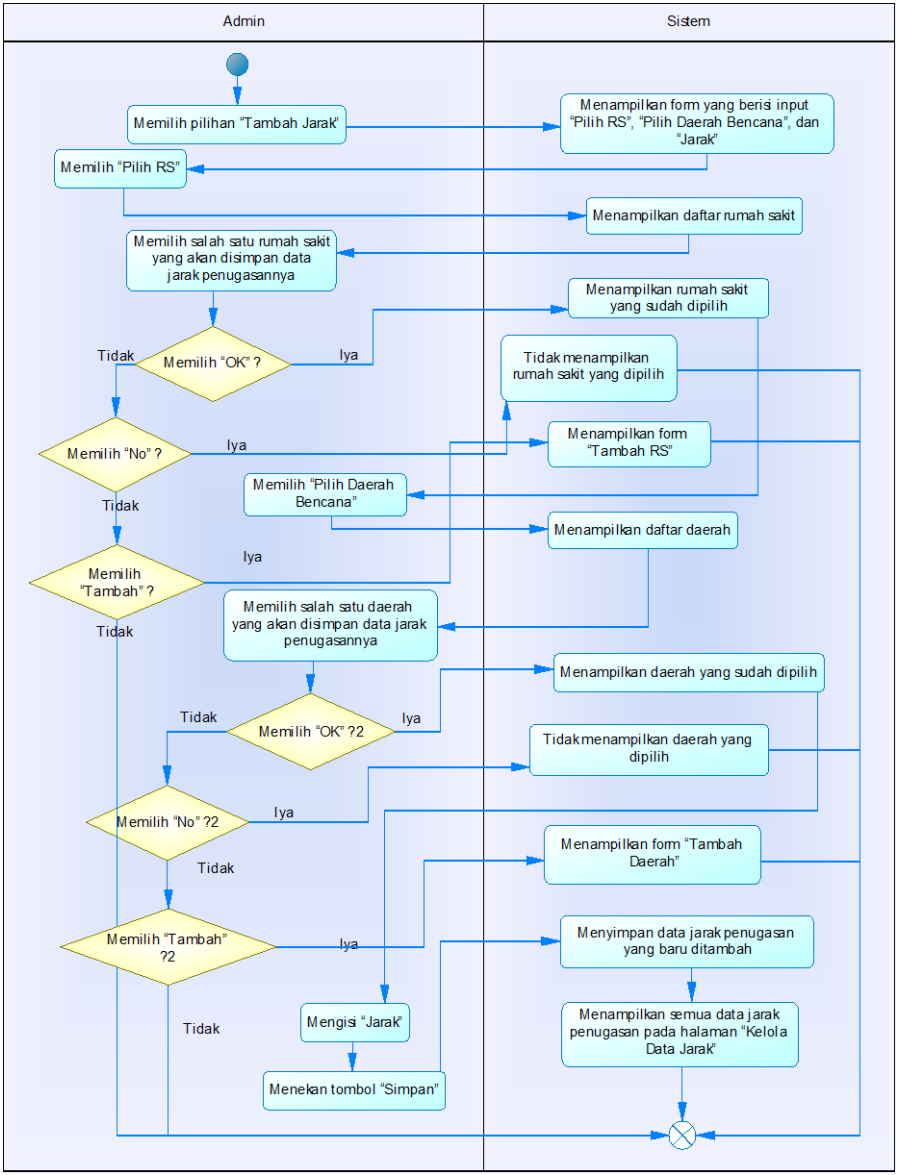
Gambar 0.15 Diagram Aktivitas UC-0015



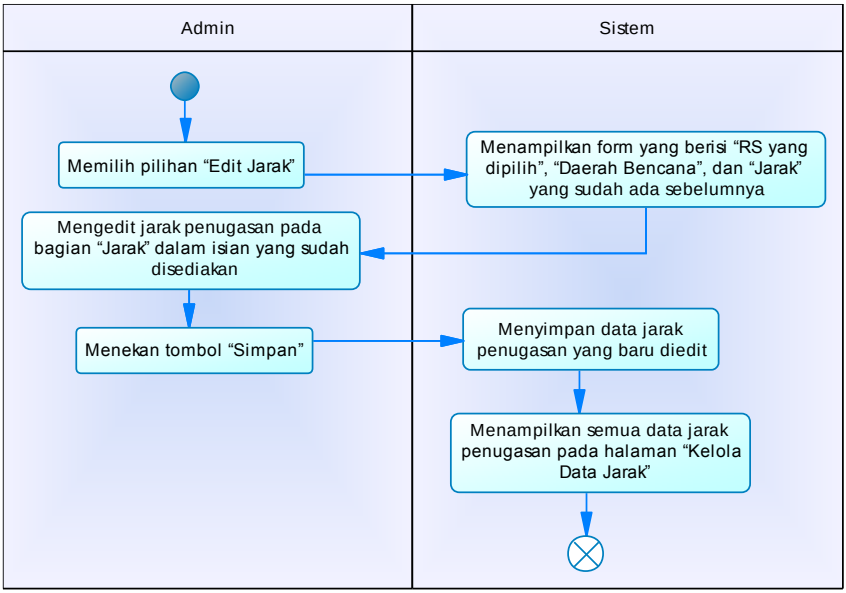
Gambar 0.16 Diagram Aktivitas UC-0016



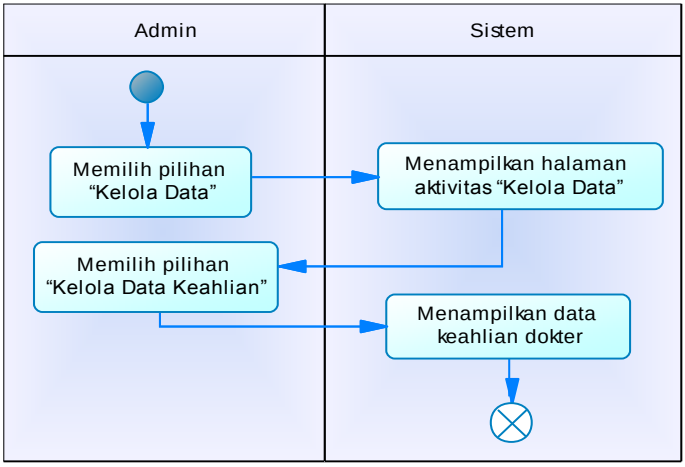
Gambar 0.17 Diagram Aktivitas UC-0017



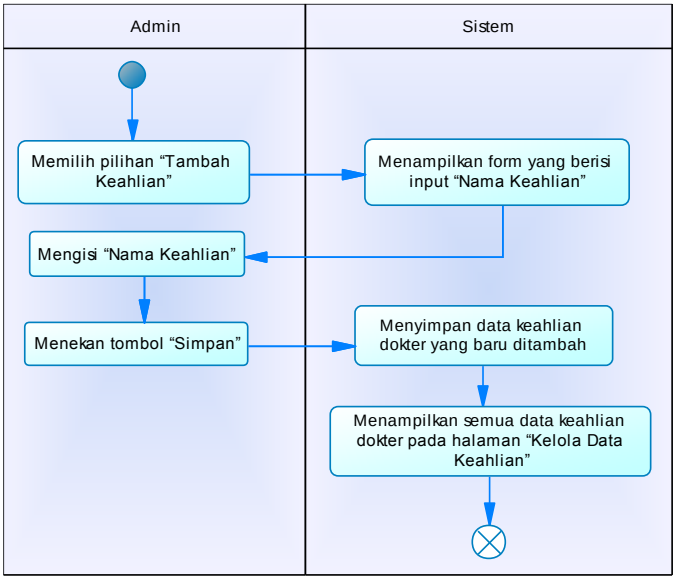
Gambar 0.18 Diagram Aktivitas UC-0018



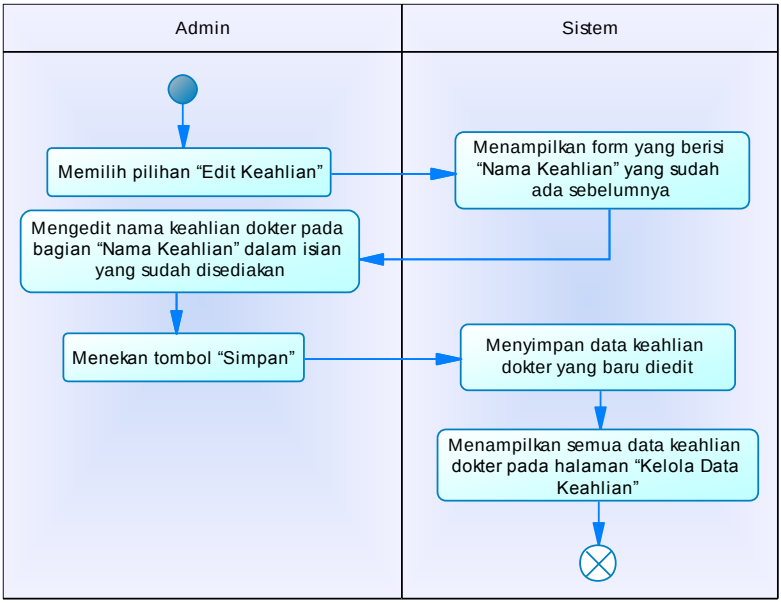
Gambar 0.19 Diagram Aktivitas UC-0019



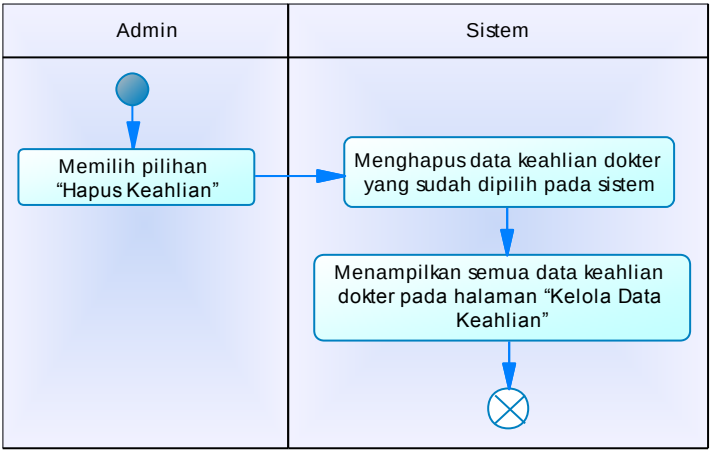
Gambar 0.20 Diagram Aktivitas UC-0020



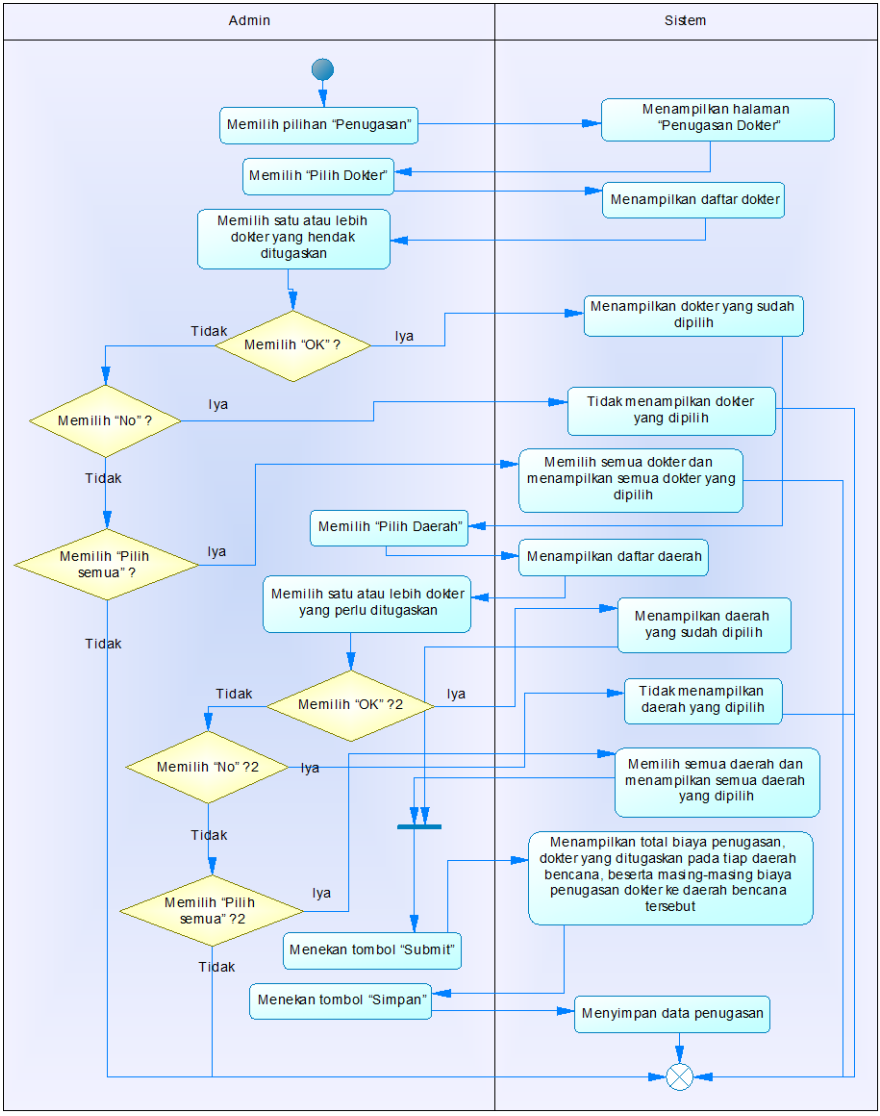
Gambar 0.21 Diagram Aktivitas UC-0021



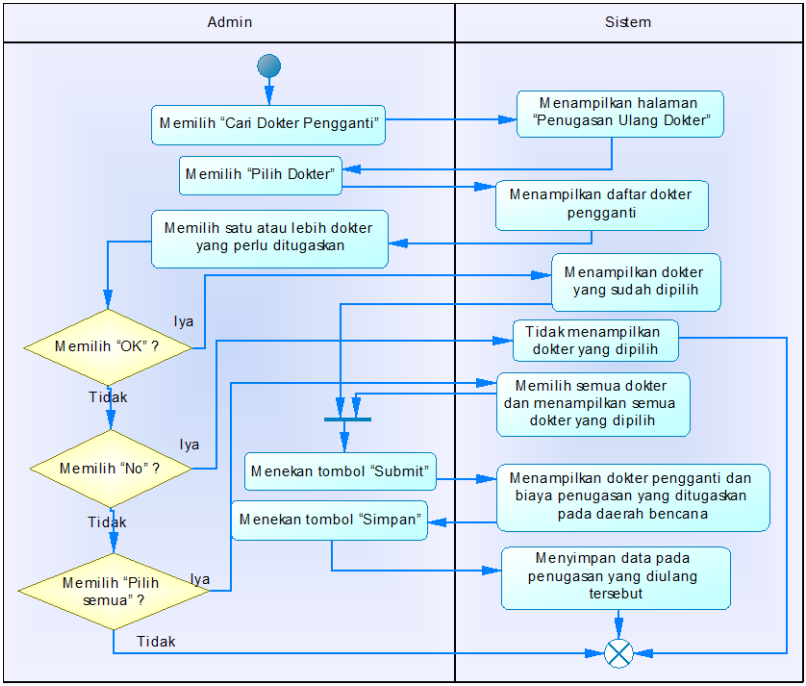
Gambar 0.22 Diagram Aktivitas UC-0022



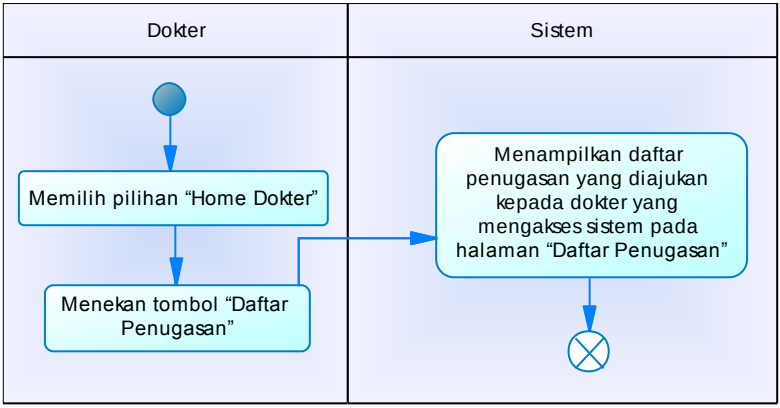
Gambar 0.23 Diagram Aktivitas UC-0023



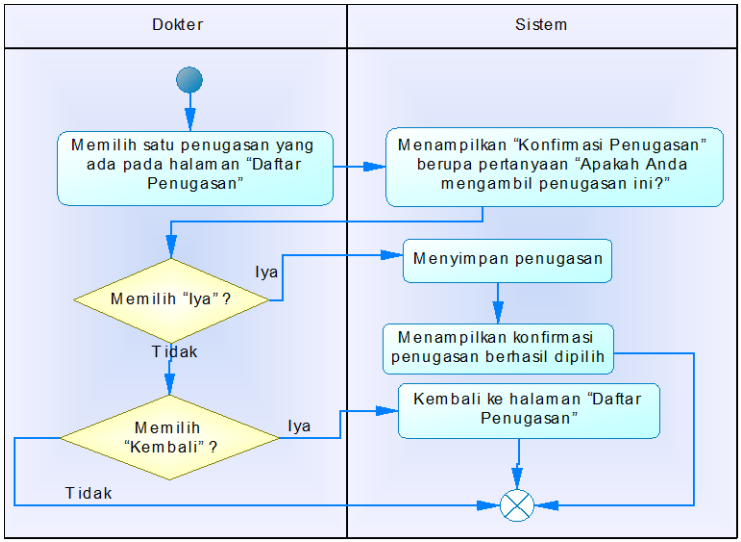
Gambar 0.24 Diagram Aktivitas UC-0024



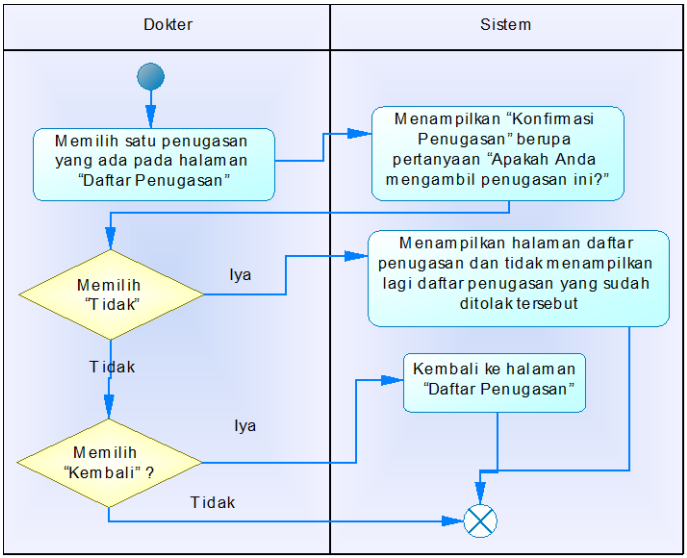
Gambar 0.25 Diagram Aktivitas UC-0025



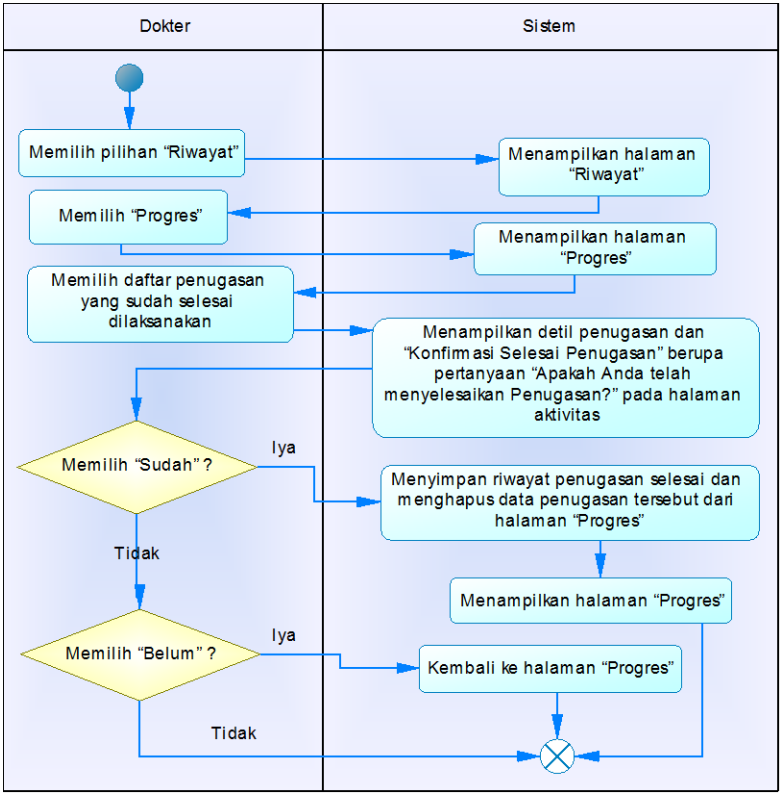
Gambar 0.26 Diagram Aktivitas UC-0026



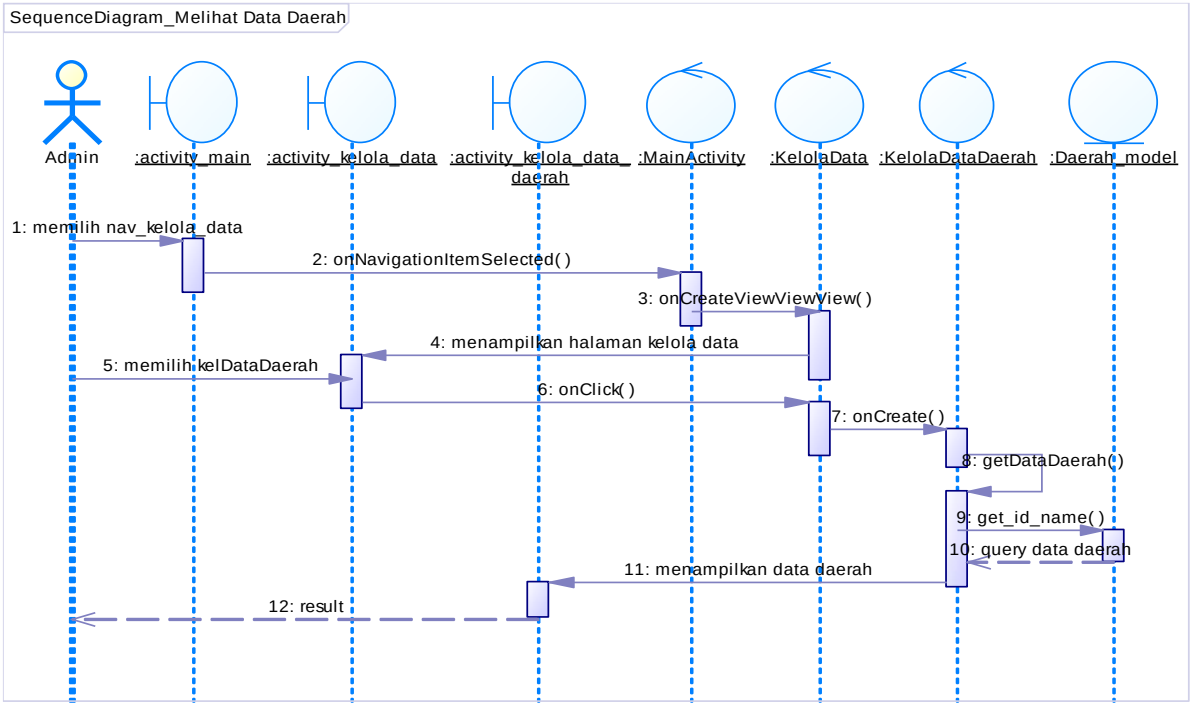
Gambar 0.27 Diagram Aktivitas UC-0027



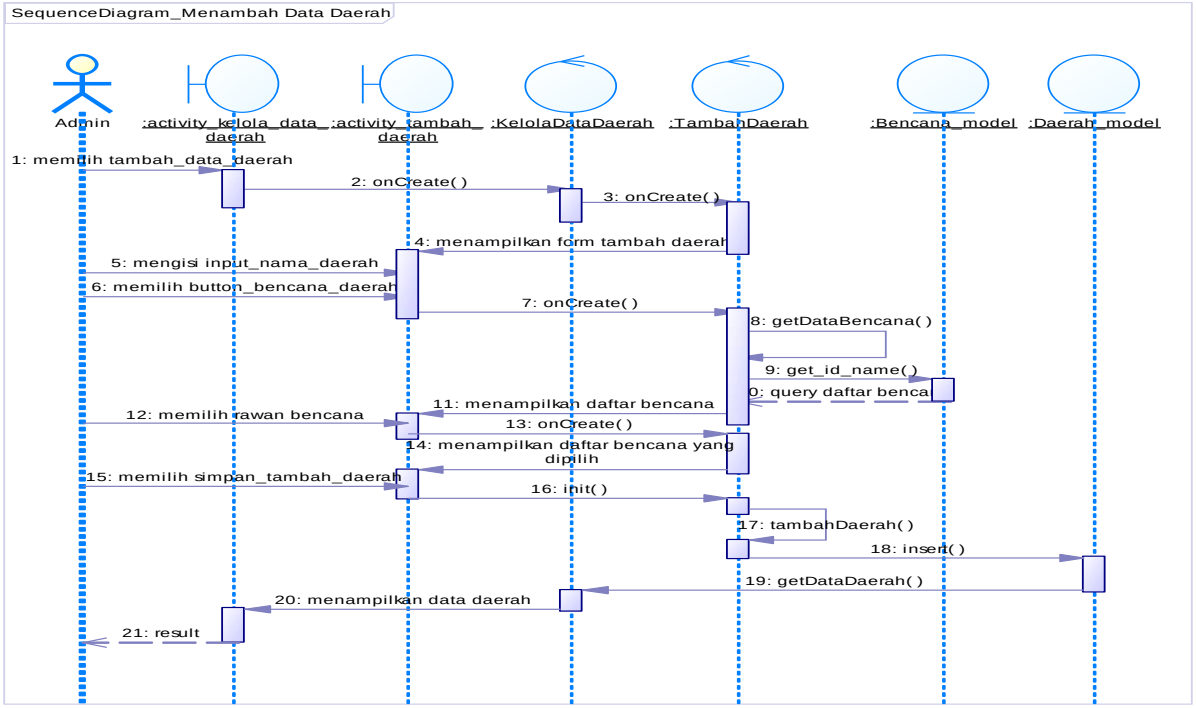
Gambar 0.28 Diagram Aktivitas UC-0028



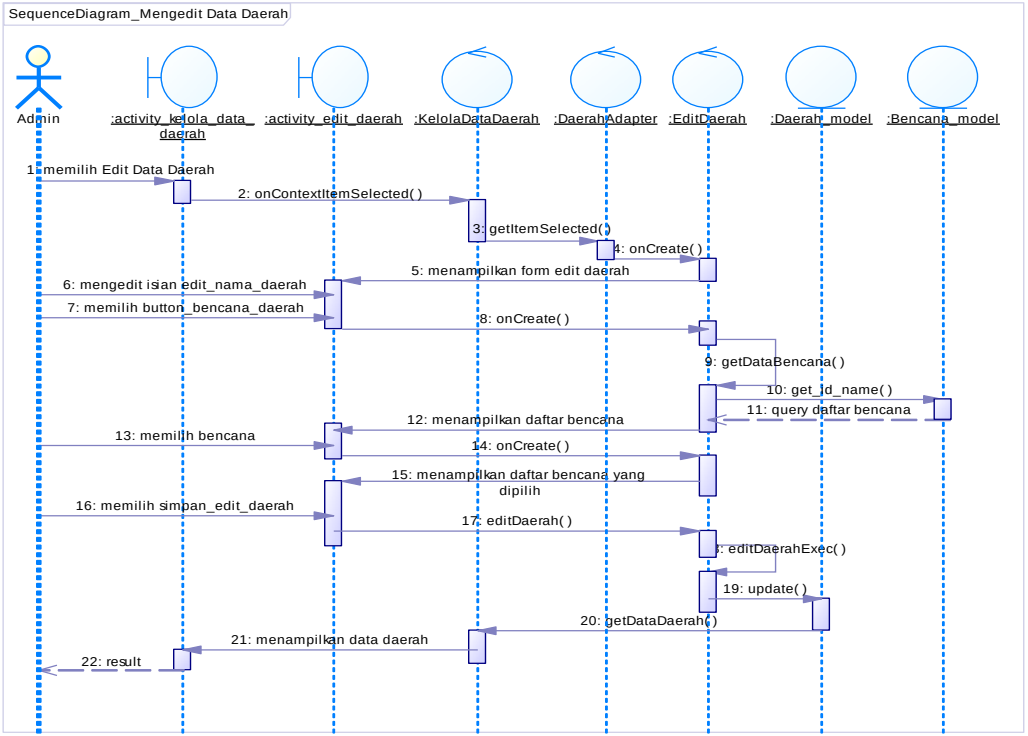
Gambar 0.29 Diagram Aktivitas UC-0029



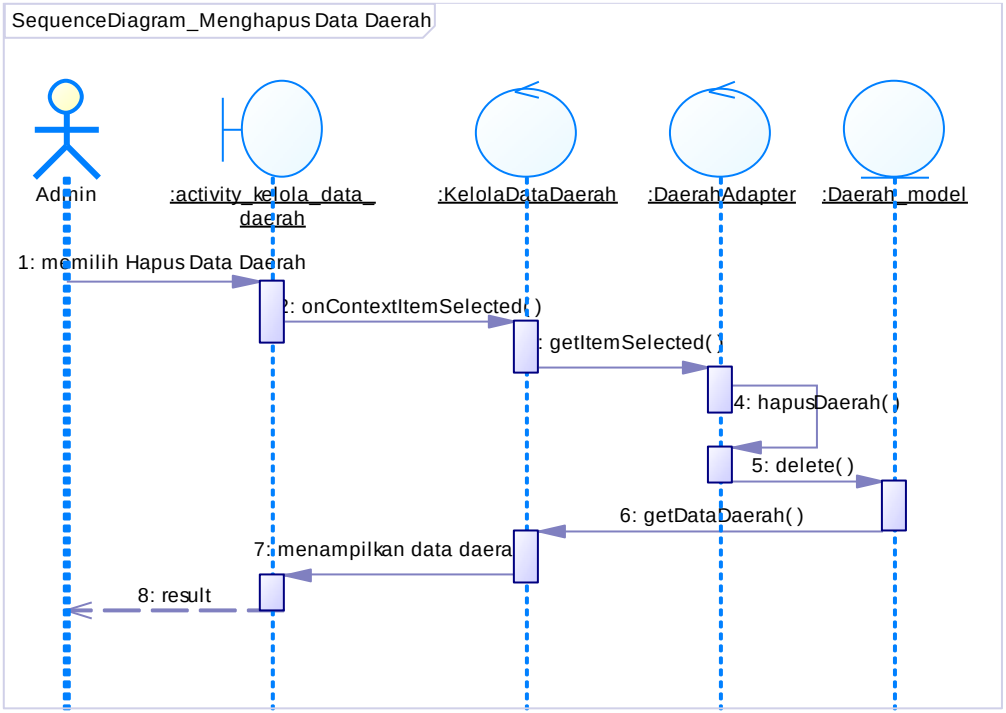
Gambar 0.30 Diagram Sekuens UC-0001



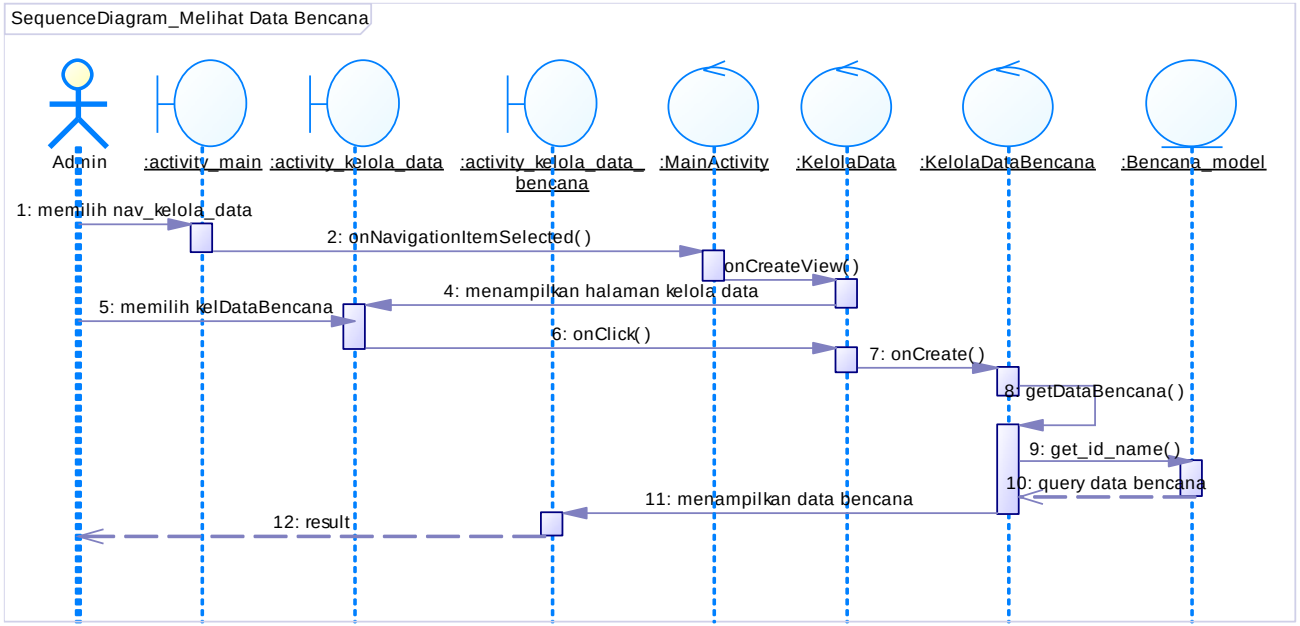
Gambar 0.31 Diagram Sekuens UC-0002



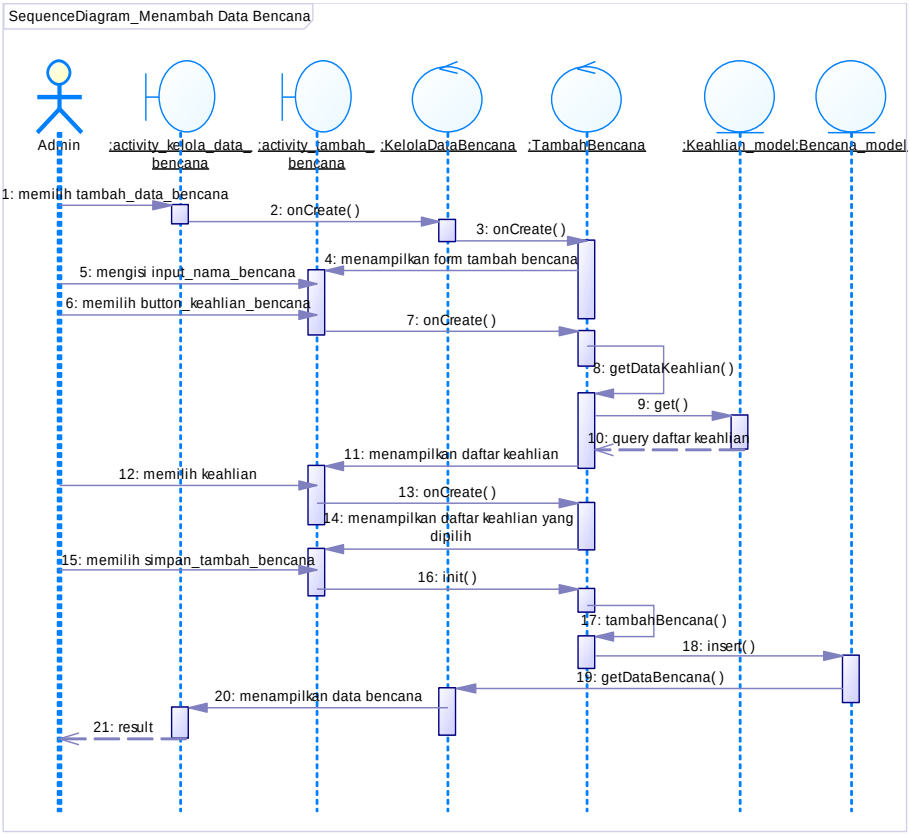
Gambar 0.32 Diagram Sekuens UC-0003



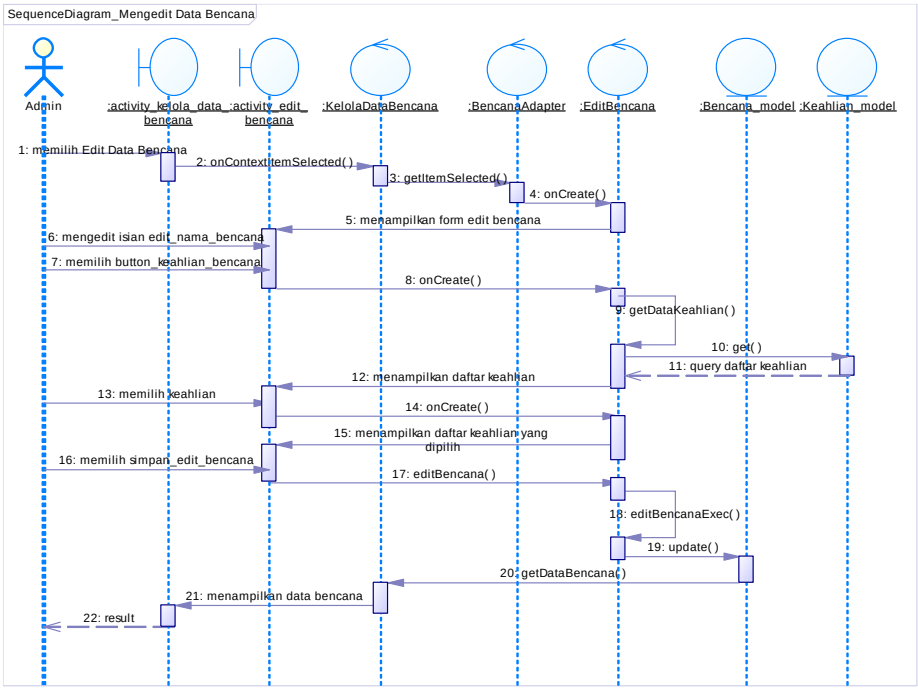
Gambar 0.33 Diagram Sekuens UC-0004



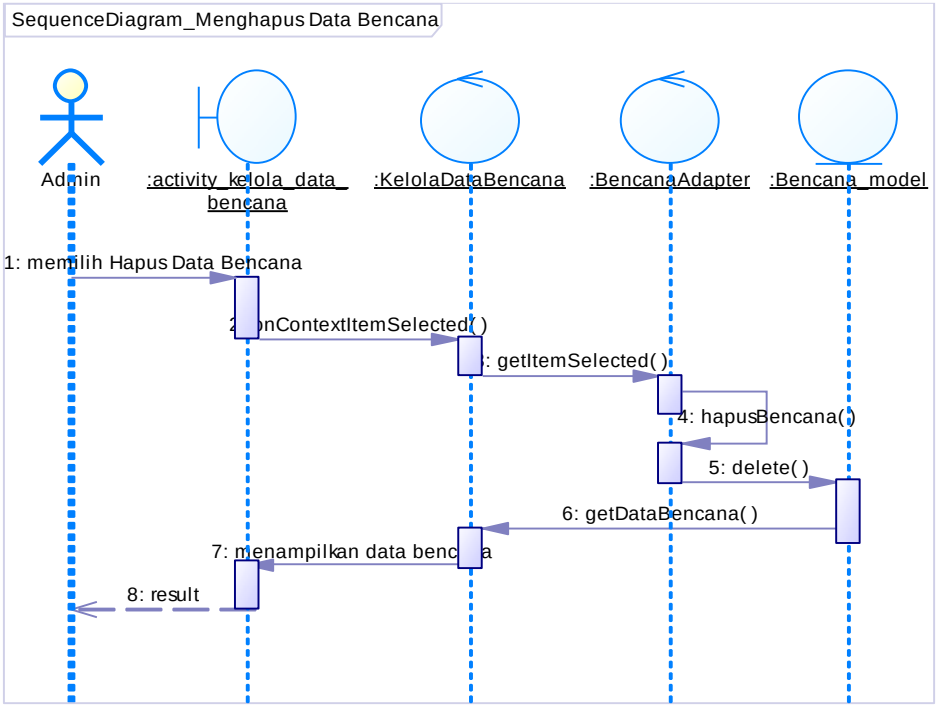
Gambar 0.34 Diagram Sekuens UC-0005



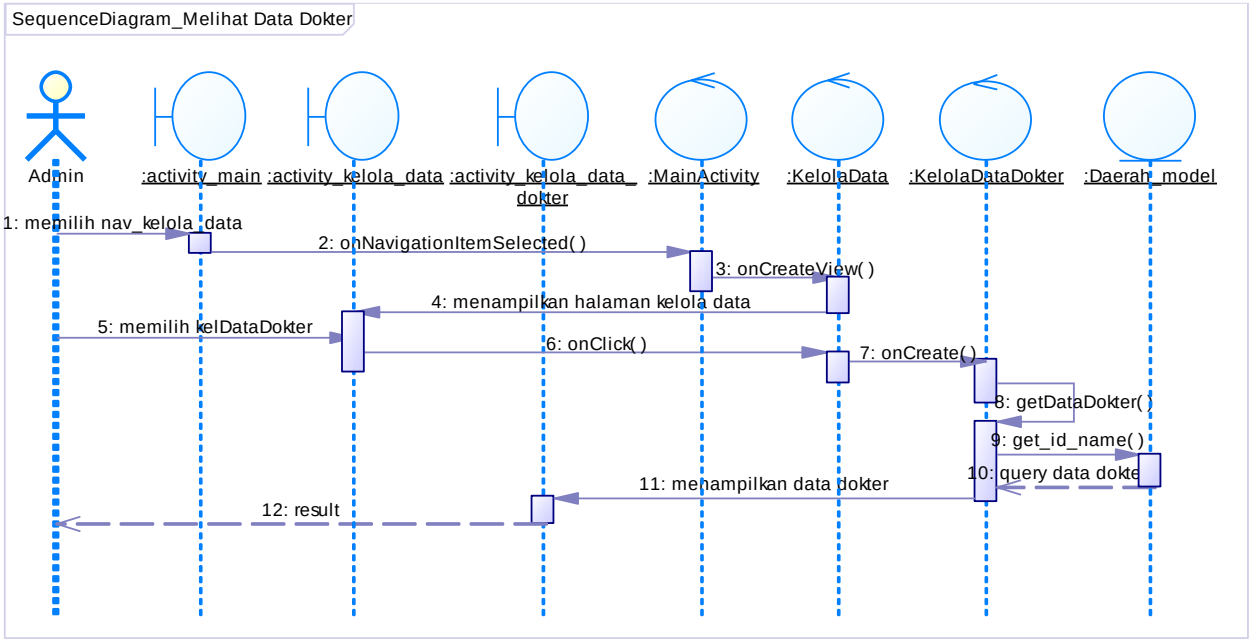
Gambar 0.35 Diagram Sekuens UC-0006



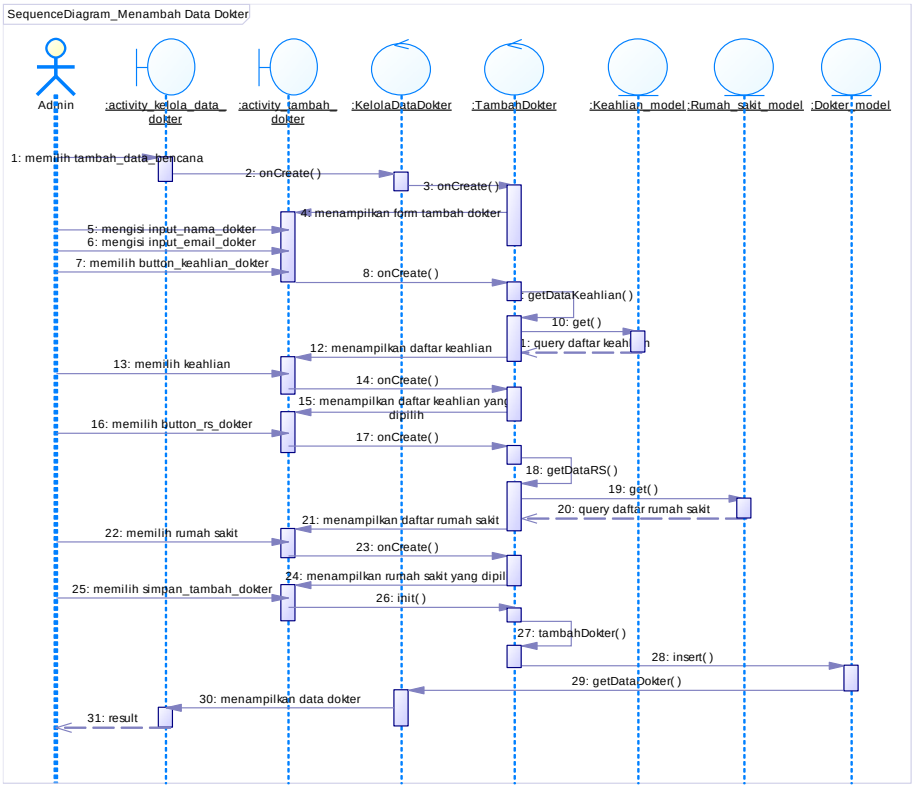
Gambar 0.36 Diagram Sekuens UC-0007



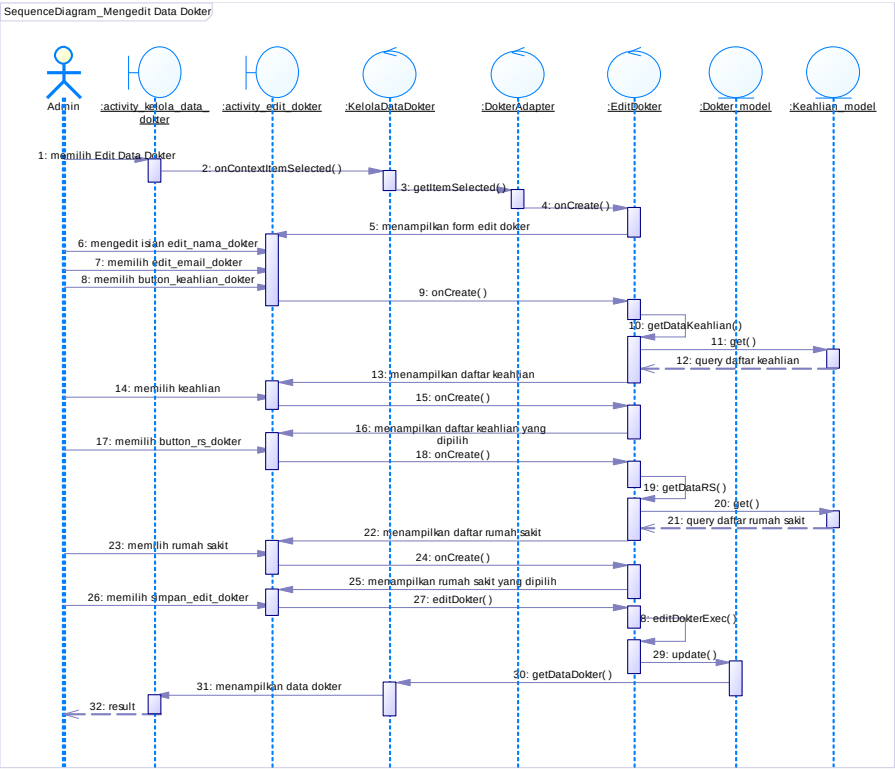
Gambar 0.37 Diagram Sekuens UC-0008



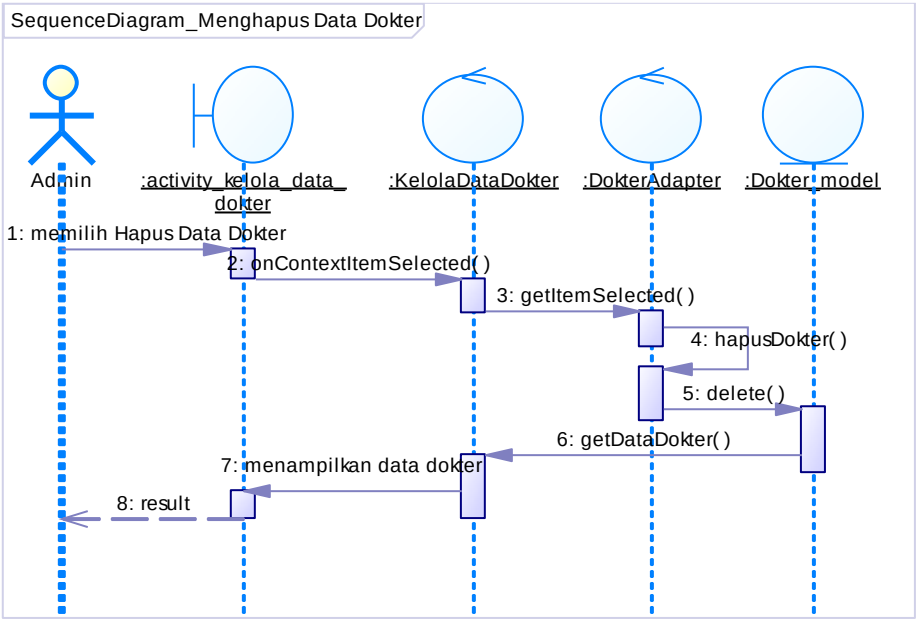
Gambar 0.38 Diagram Sekuens UC-0009



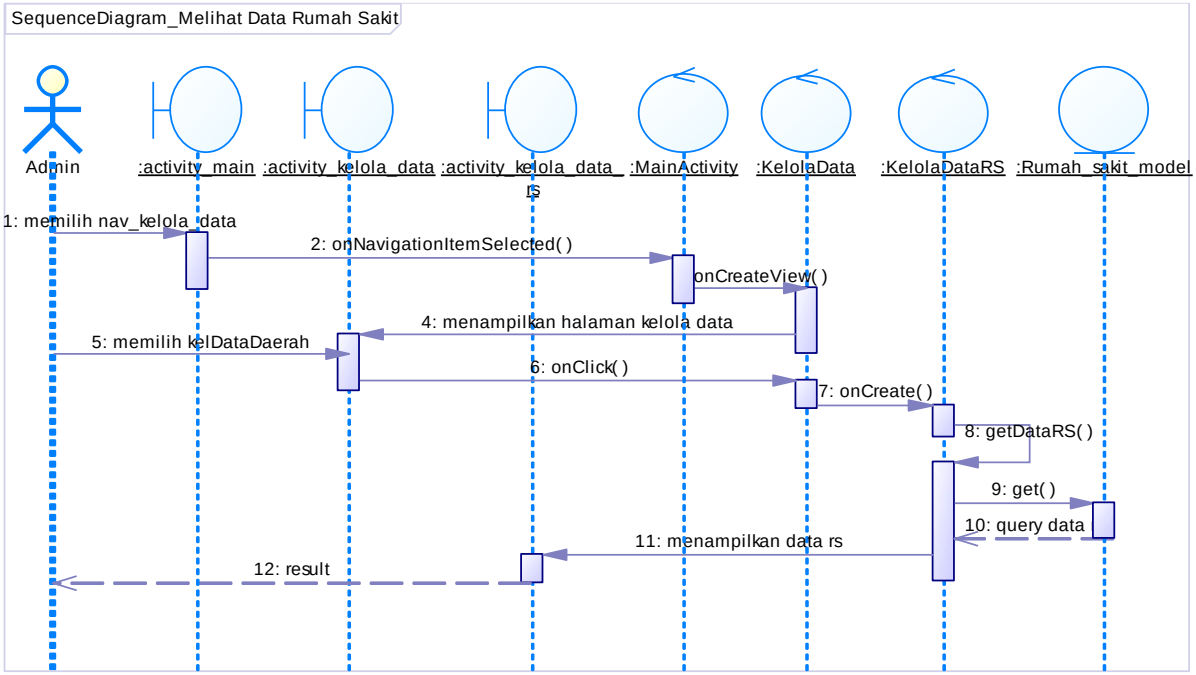
Gambar 0.39 Diagram Sekuens UC-0010



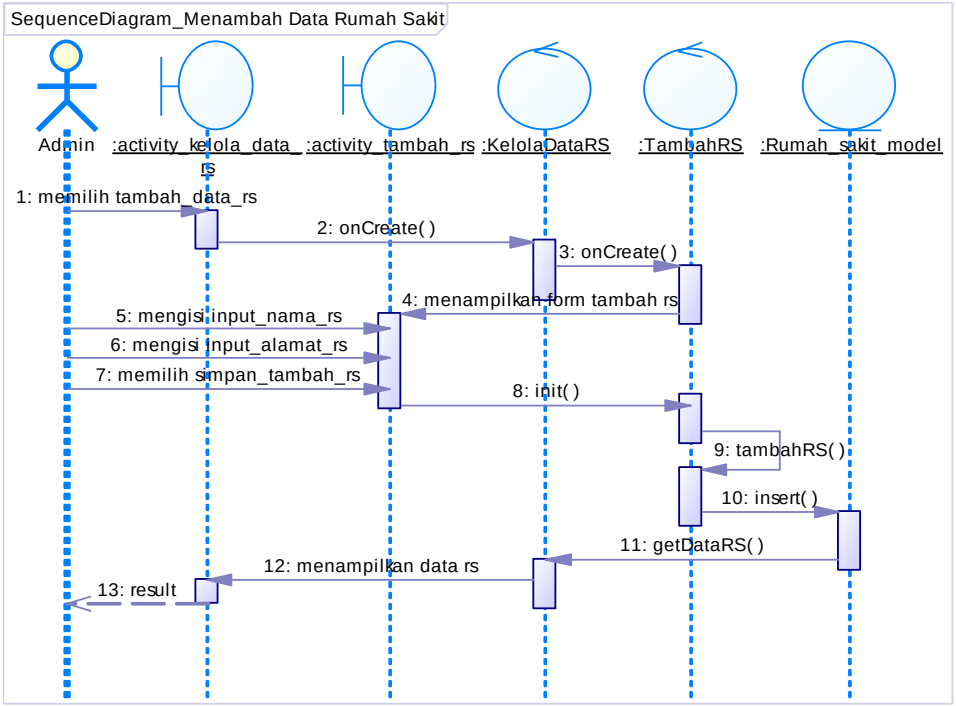
Gambar 0.40 Diagram Sekuens UC-0011



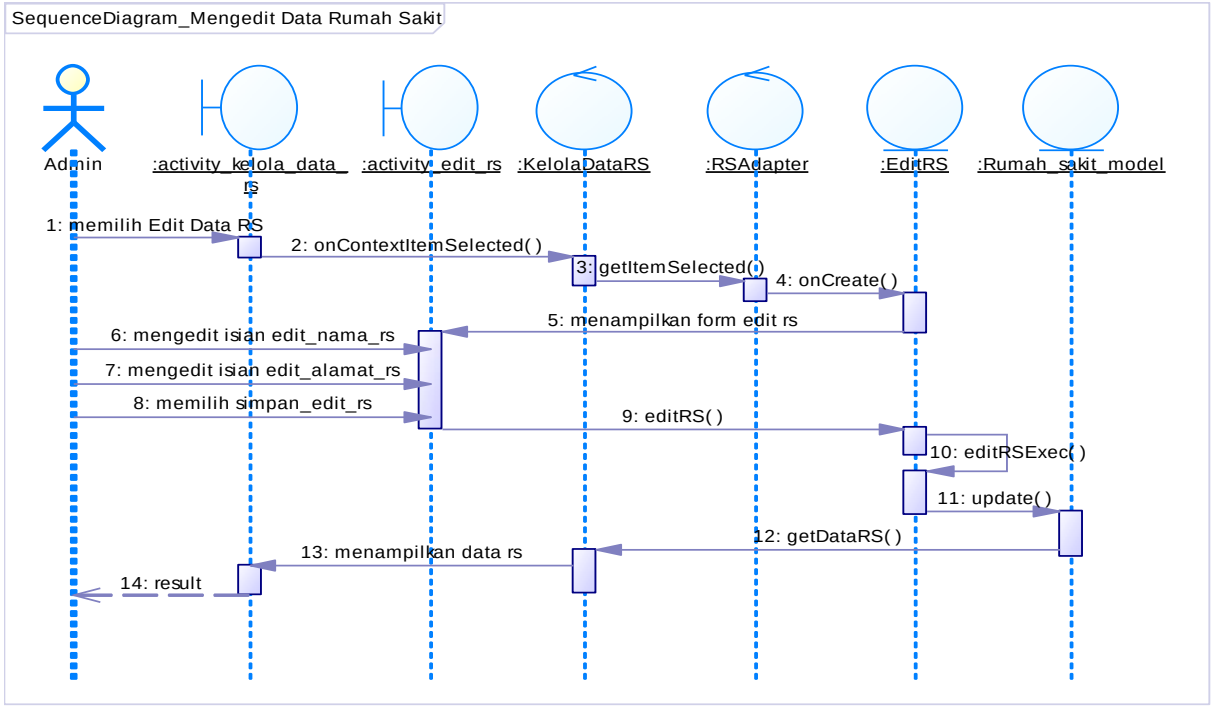
Gambar 0.41 Diagram Sekuens UC-0012



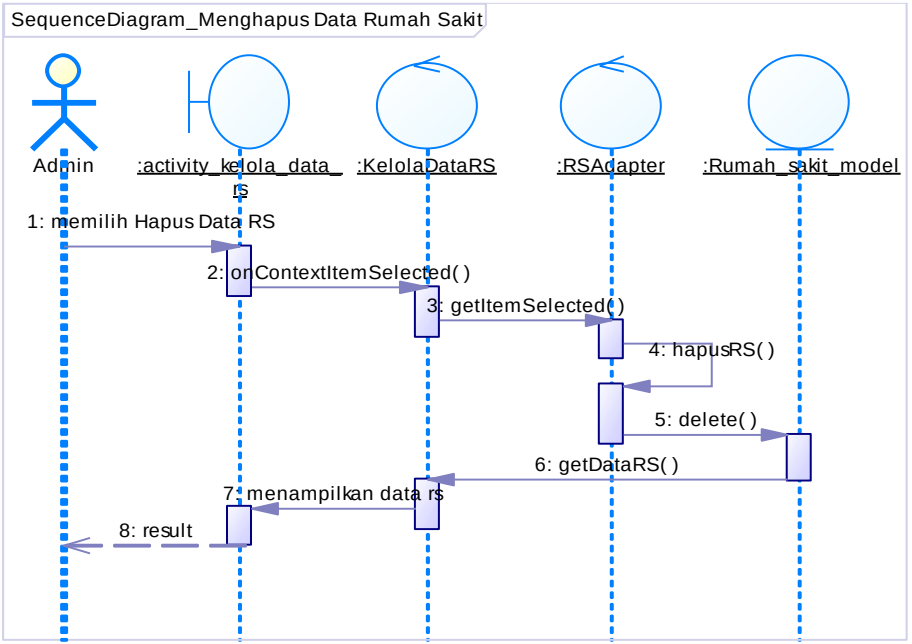
Gambar 0.42 Diagram Sekuens UC-0013



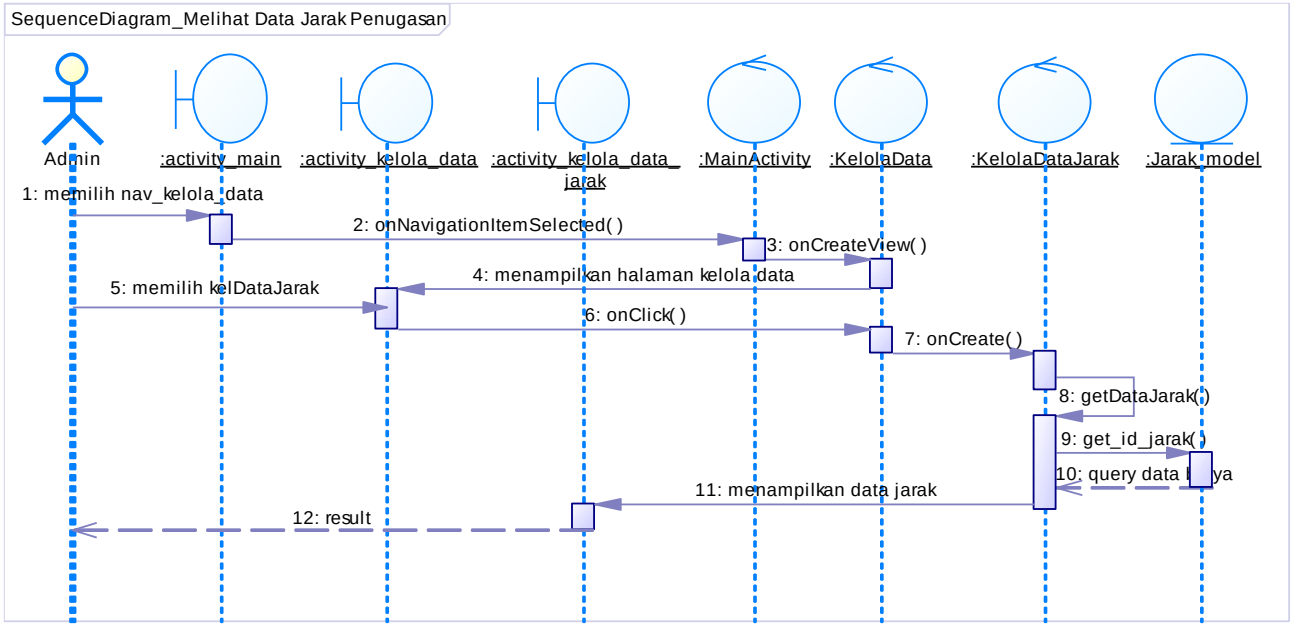
Gambar 0.43 Diagram Sekuens UC-0014



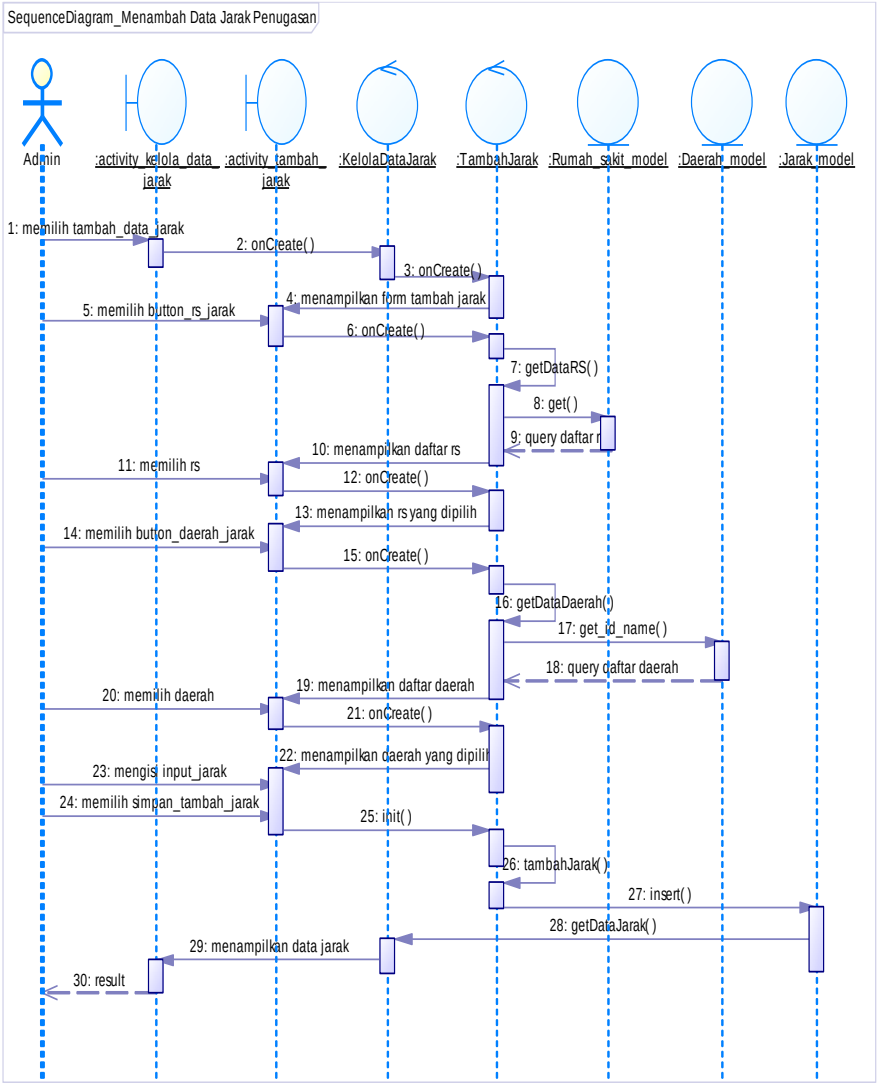
Gambar 0.44 Diagram Sekuens UC-0015



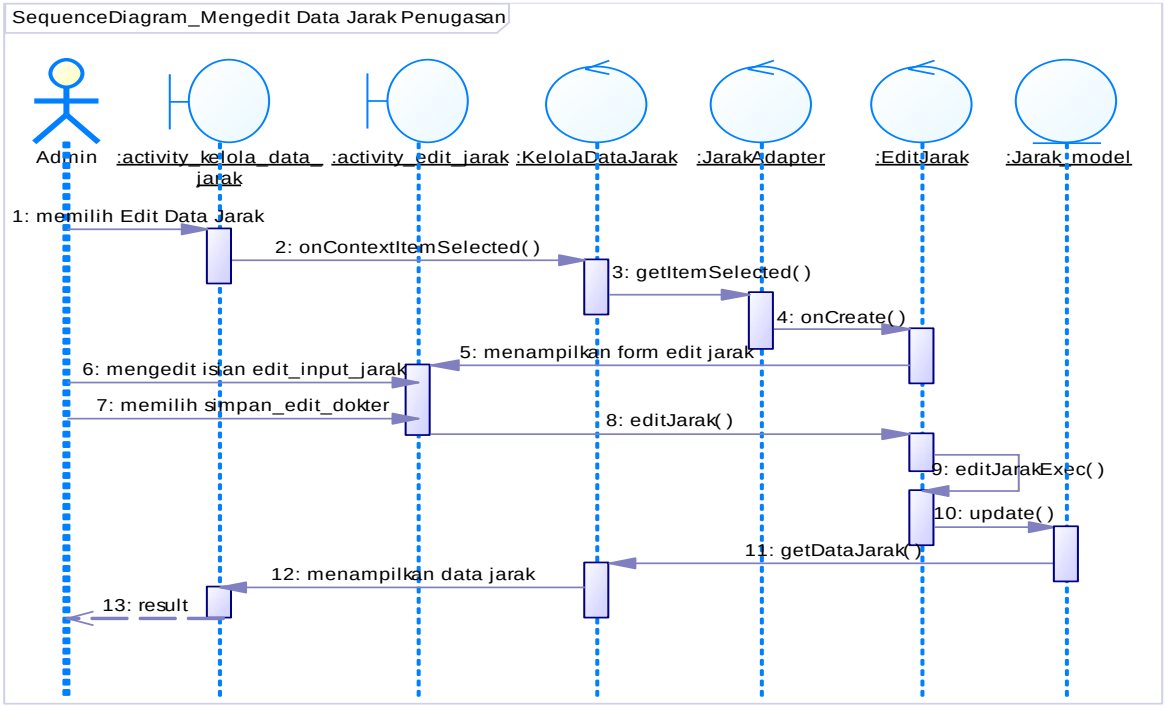
Gambar 0.45 Diagram Sekuens UC-0016



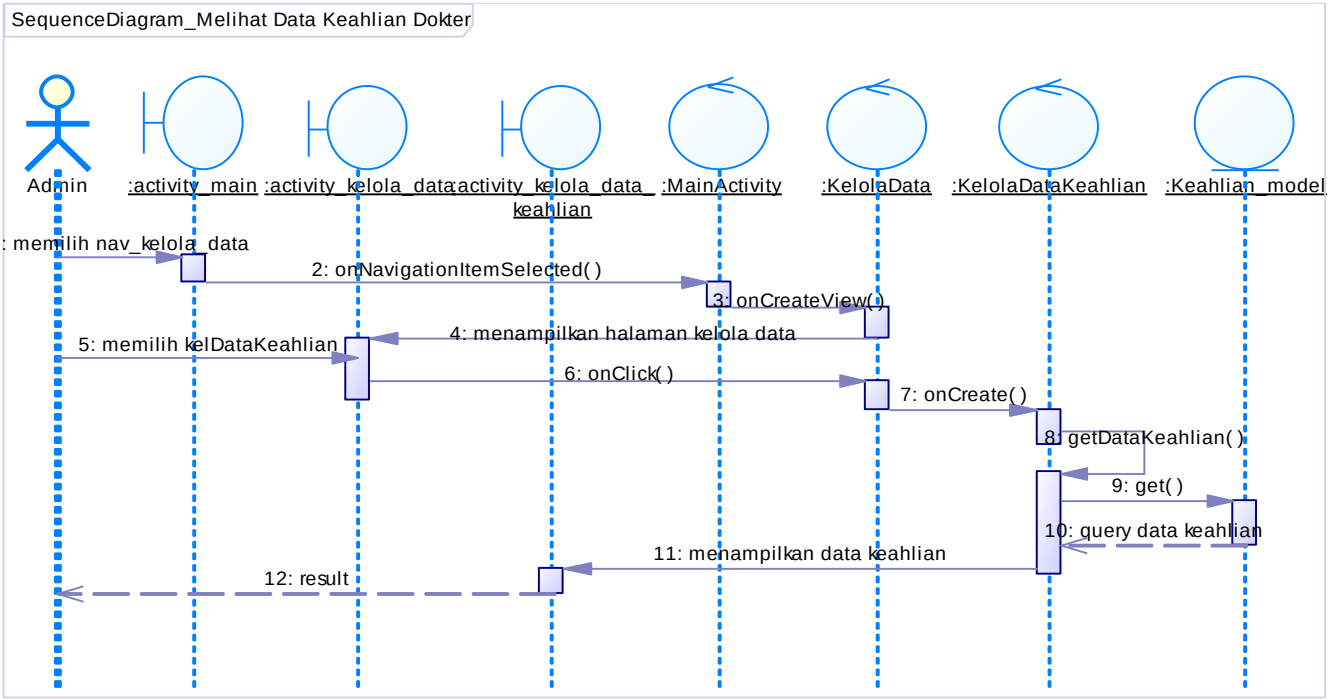
Gambar 0.46 Diagram Sekuens UC-0017



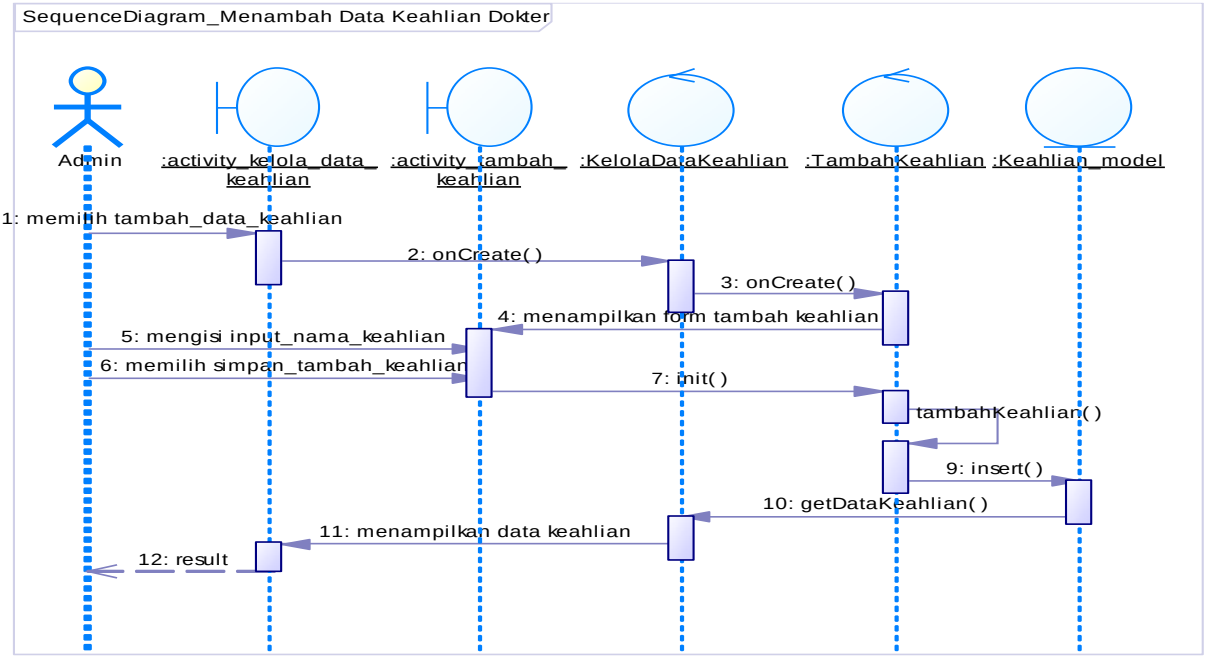
Gambar 0.47 Diagram Sekuens UC-0018



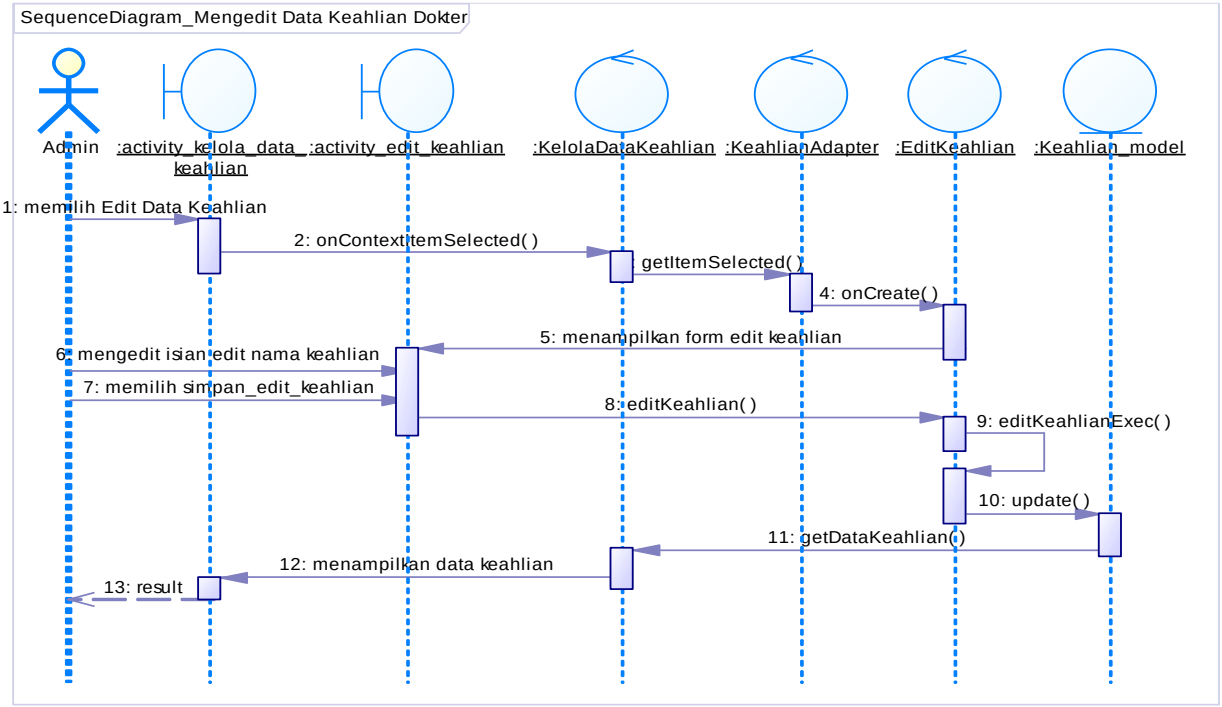
Gambar 0.48 Diagram Sekuens UC-0019



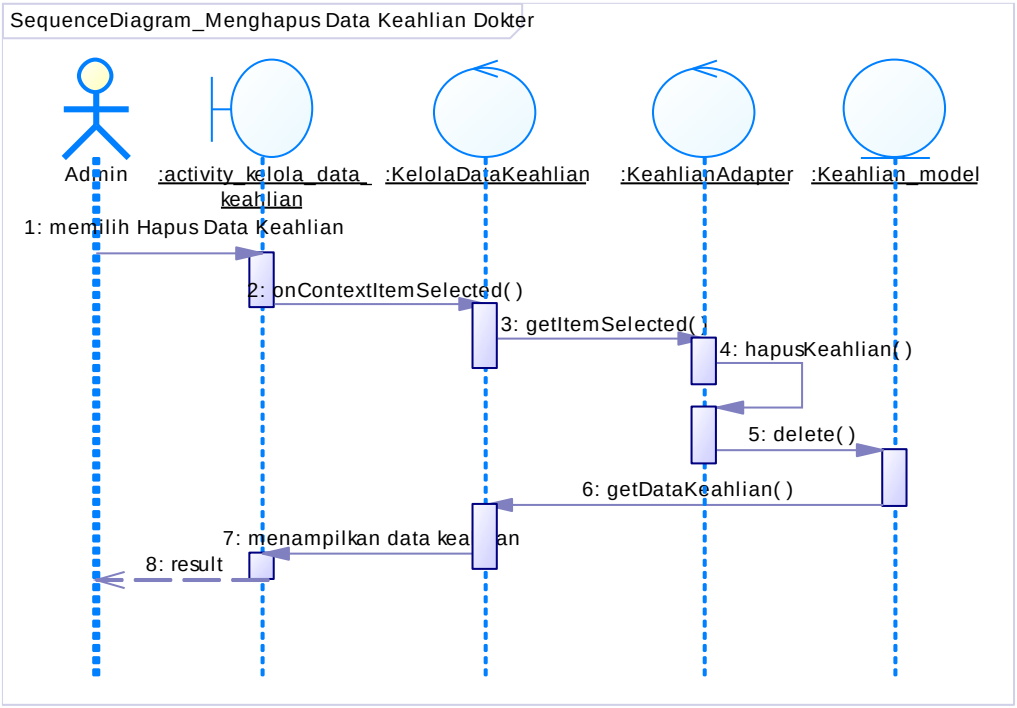
Gambar 0.49 Diagram Sekuens UC-0020



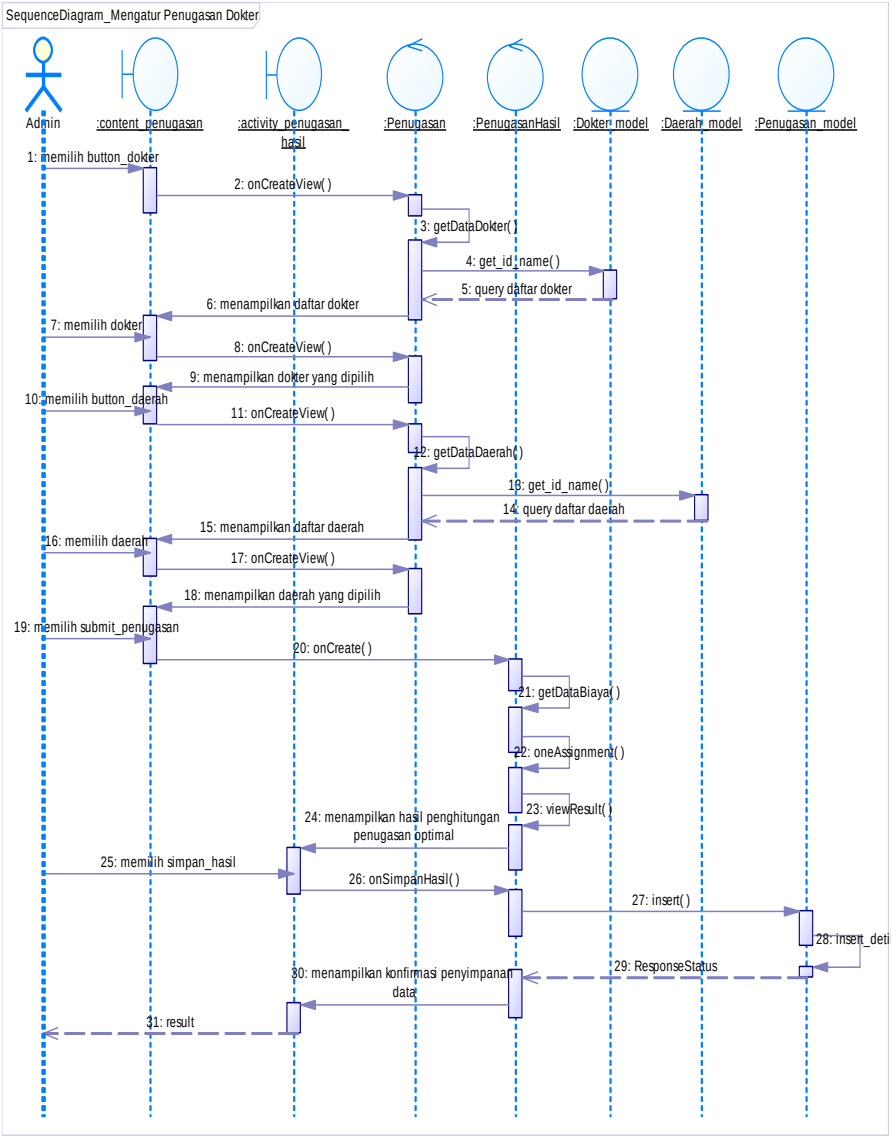
Gambar 0.50 Diagram Sekuens UC-0021



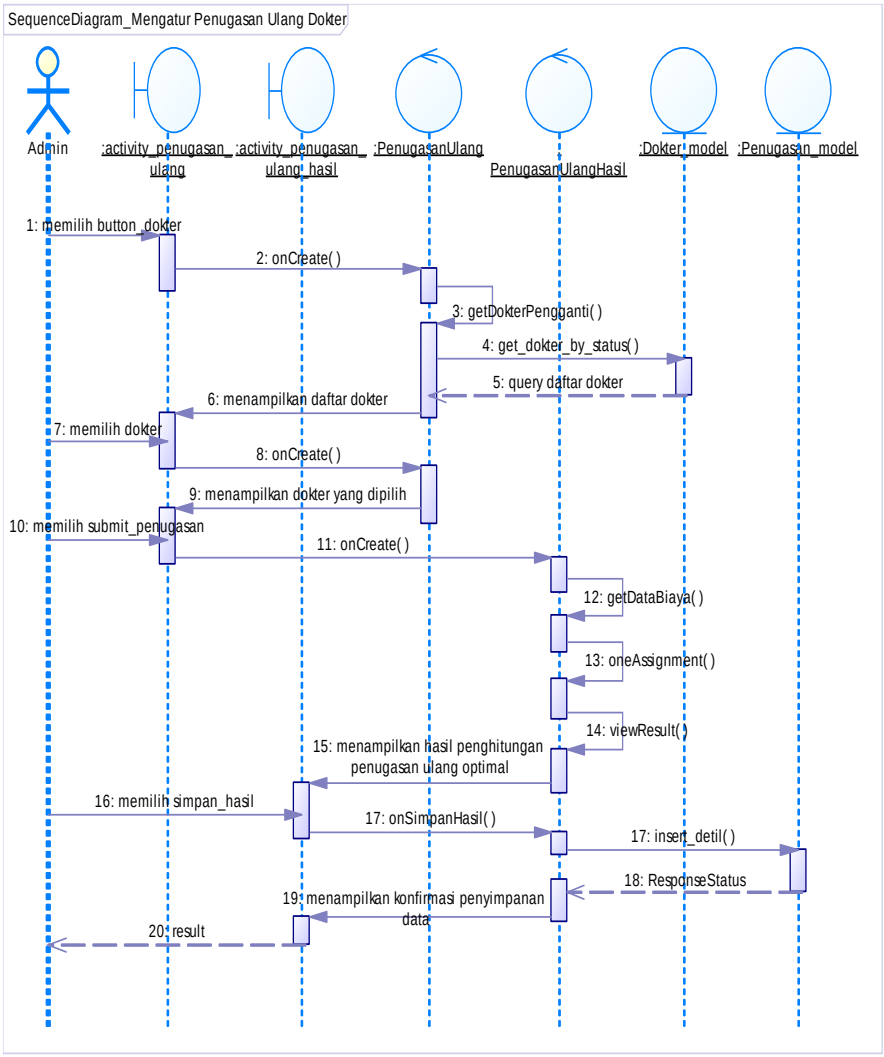
Gambar 0.51 Diagram Sekuens UC-0022



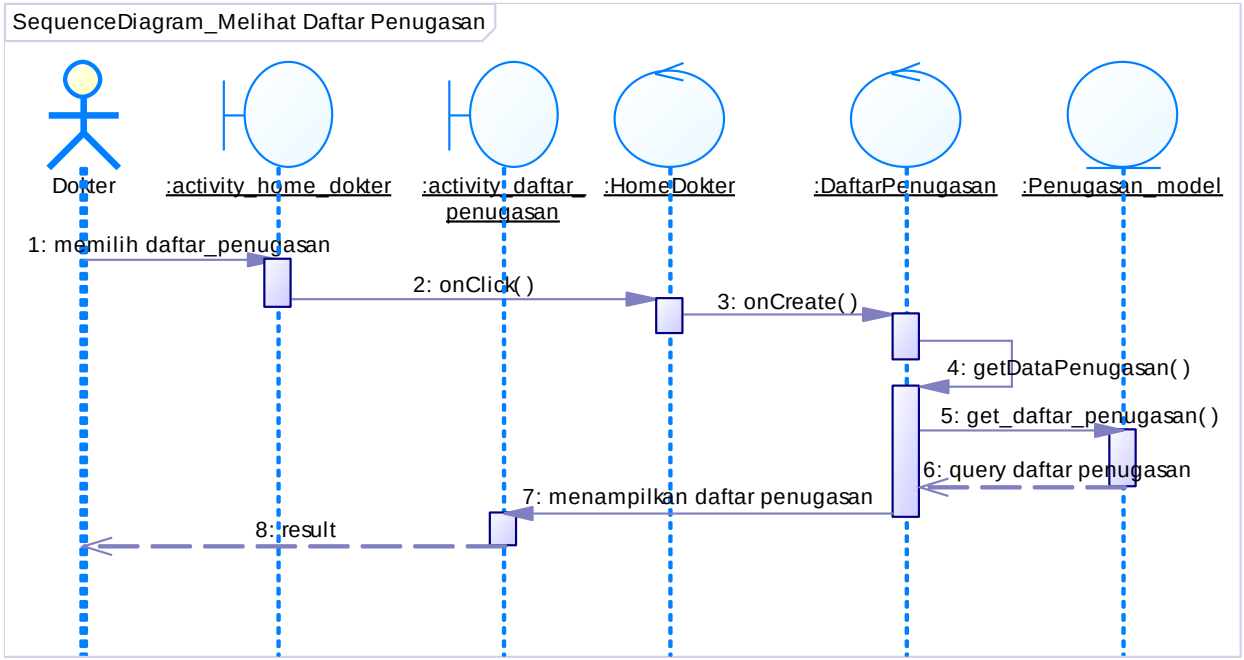
Gambar 0.52 Diagram Sekuens UC-0023



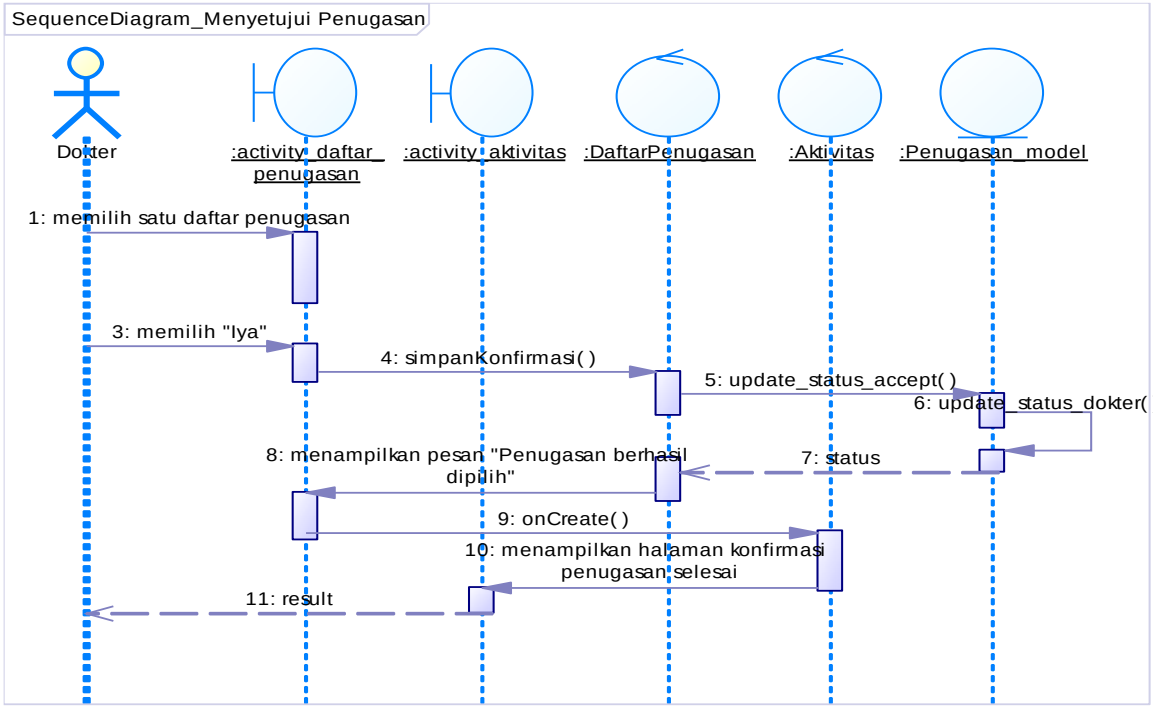
Gambar 0.53 Diagram Sekuens UC-0024



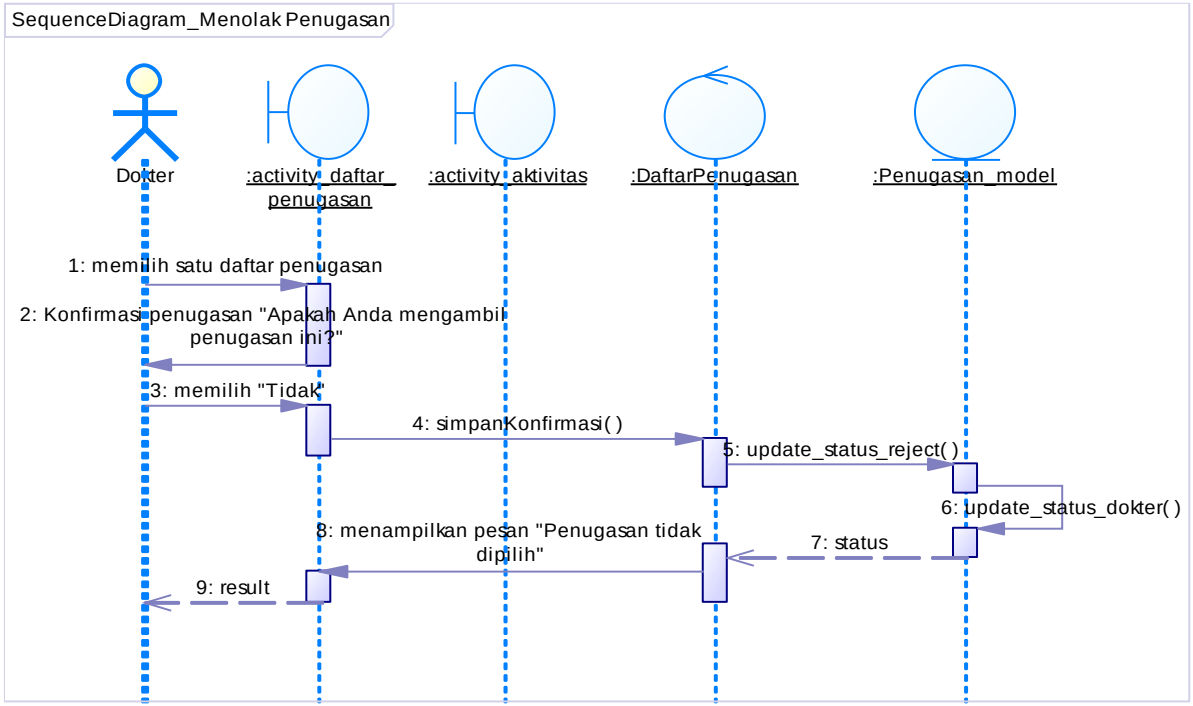
Gambar 0.54 Diagram Sekuens UC-0025



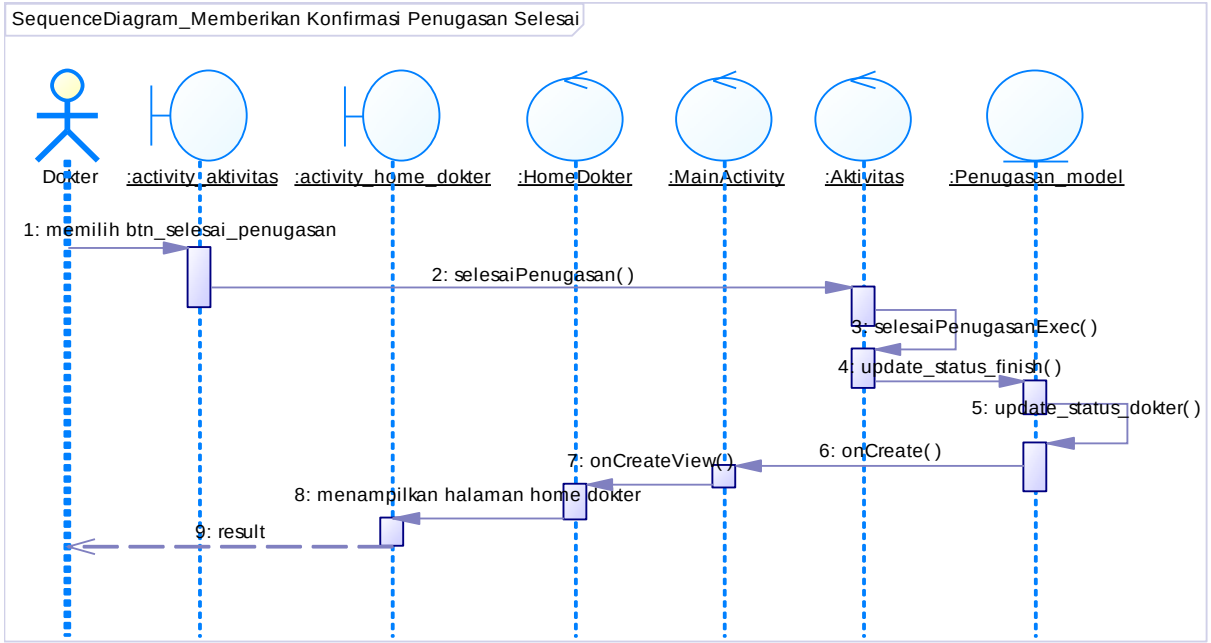
Gambar 0.55 Diagram Sekuens UC-0026



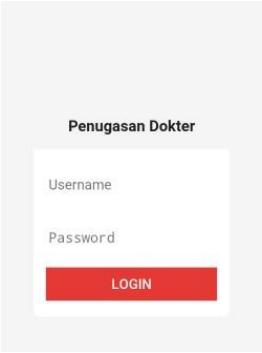
Gambar 0.56 Diagram Sekuens UC-0027



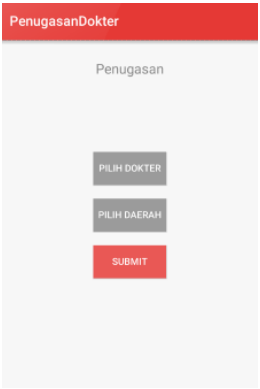
Gambar 0.57 Diagram Sekuens UC-0028



Gambar 0.58 Diagram Sekuens UC-0029



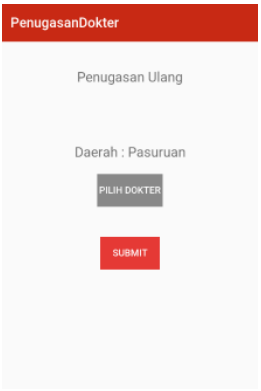
Gambar 0.59 Rancangan Halaman Antarmuka Login Pengguna



Gambar 0.60 Rancangan Halaman Antarmuka Penugasan

Gambar 0.61 Rancangan Halaman Antarmuka Penugasan Hasil

Gambar 0.62 Rancangan Halaman Antarmuka Riwayat Progres Penugasan Admin



Gambar 0.63 Rancangan Halaman Antarmuka Penugasan Ulang

Gambar 0.64 Rancangan Halaman Antarmuka Daftar Penugasan

Gambar 0.65 Rancangan Halaman Antarmuka Riwayat Progres Penugasan Dokter

Gambar 0.66 Rancangan Halaman Antarmuka Konfirmasi Penugasan Selesai

Tabel 0.1 Data Uji Jarak antara Rumah Sakit ke Daerah

No	Rumah Sakit	Daerah	Jarak (km)
1	Adi Husada Kapasari Surabaya (Jl. Kapasari 97-101 Surabaya, Jawa Timur)	Surabaya	3
		Mojokerto	62
		Sidoarjo	34
		Gresik	24
		Pasuruan	100
		Nganjuk	131
		Bojonegoro	133
		Kediri	138
		Jember	203
		Tulungagung	177
2	Darmo Surabaya (Jl. Raya Darmo No.90, Surabaya)	Surabaya	5
		Mojokerto	48
		Sidoarjo	31
		Gresik	22
		Pasuruan	64
		Nganjuk	115
		Bojonegoro	133
		Kediri	122
		Jember	194
		Tulungagung	157
3	Umum Panti Nirmala Malang (Jl. Kebalen Wetan, Kota Malang)	Surabaya	98
		Mojokerto	89
		Sidoarjo	72
		Gresik	111
		Pasuruan	54
		Nganjuk	122
		Bojonegoro	222
		Kediri	112
		Jember	185
		Tulungagung	99
4	Permata Bunda Malang (Jl. Soekarno Hatta No. 75, Kota Malang)	Surabaya	94
		Mojokerto	85
		Sidoarjo	68
		Gresik	107

		Pasuruan	51
		Nganjuk	116
		Bojonegoro	218
		Kediri	98
		Jember	181
		Tulungagung	106
5	Alhuda Banyuwangi (Jl. Diponegoro No. 65 Genteng Banyuwangi)	Surabaya	270
		Mojokerto	267
		Sidoarjo	243
		Gresik	282
		Pasuruan	205
		Nganjuk	332
		Bojonegoro	394
		Kediri	339
		Jember	73
		Tulungagung	320
6	Dr Harjono Ponorogo (Jl. Raya Ponorogo – Pacitan Ponorogo, Ponorogo)	Surabaya	204
		Mojokerto	152
		Sidoarjo	181
		Gresik	206
		Pasuruan	211
		Nganjuk	81
		Bojonegoro	101
		Kediri	108
		Jember	343
		Tulungagung	86
7	Dr. R. Koesman Tuban (Jl. Dr. Wahidin Sudiro Husodo No.800 Tuban)	Surabaya	101
		Mojokerto	100
		Sidoarjo	127
		Gresik	85
		Pasuruan	159
		Nganjuk	123
		Bojonegoro	77
		Kediri	132
		Jember	291
		Tulungagung	165
8	Dr. Soebandi Jember (Jl. Dr.	Surabaya	207

	Soebandi No.124, Jember)	Mojokerto	204
		Sidoarjo	180
		Gresik	219
		Pasuruan	142
		Nganjuk	269
		Bojonegoro	331
		Kediri	276
		Jember	9
		Tulungagung	257
9	RS Bhayangkara Nganjuk (Jl. AR. Saleh No. 56 Nganjuk)	Surabaya	125
		Mojokerto	73
		Sidoarjo	102
		Gresik	127
		Pasuruan	132
		Nganjuk	2
		Bojonegoro	60
		Kediri	36
		Jember	263
		Tulungagung	70
10	RS Anak & Bunda Bojonegoro (Jl. K.H Hasyim Asy'ari No 17 Bojonegoro)	Surabaya	111
		Mojokerto	100
		Sidoarjo	131
		Gresik	95
		Pasuruan	169
		Nganjuk	64
		Bojonegoro	1
		Kediri	101
		Jember	301
		Tulungagung	135

BIODATA PENULIS

Qonita Luthfia Sutino, lahir di Jakarta, pada tanggal 13 Oktober 1993. Penulis menempuh pendidikan SMP di SMP Negeri 12 Kota Bogor, SMA di SMA Negeri 3 Kota Bogor, dan S1 Teknik Informatika Fakultas Teknologi Informasi di Institut Teknologi Sepuluh Nopember Surabaya (2013-2017).

Selama masa kuliah, penulis aktif dalam organisasi Himpunan Mahasiswa Teknik Computer (HMTC), di antaranya adalah menjadi staf Departemen Pengembangan Profesi HMTC ITS 2014-2015, panitia Biro Hubungan Masyarakat Schematics HMTC ITS 2014, staf ahli Pengembangan Profesi HMTC ITS 2015-2016, dan wakil ketua Biro Hubungan Masyarakat Schematics HMTC ITS 2015.

Selama kuliah di Teknik Informatika ITS, penulis mengambil bidang minat Rekayasa Perangkat Lunak (RPL) dan menjadi administrator di Laboratorium Rekayasa Perangkat Lunak dengan ketertarikan penulis terdapat pada analisis perancangan sistem, arsitektur perangkat lunak, dan penjaminan mutu perangkat lunak. Penulis dapat dihubungi melalui alamat surel **qonitaluthfia@gmail.com**.